ThinApp-Benutzerhandbuch

ThinApp 4.0.4

DE-000117-04

vmware®

Die neueste technische Dokumentation finden Sie auf der Website von VMware unter:

http://www.vmware.com/support/

Die Website von VMware bietet Ihnen auch die neuesten Produkt-Updates.

Falls Sie Anmerkungen zu dieser Dokumentation haben, senden Sie diese bitte an:

docfeedback@vmware.com

Copyright © 2009 VMware, Inc. Alle Rechte vorbehalten. Dieses Produkt ist durch das US-amerikanische und internationale Urheberrecht und geistige Eigentumsrecht geschützt. Die Produkte von VMware sind durch ein oder mehrere Patente geschützt. Diese sind aufgeführt unter http://www.vmware.com/go/patents.

VMware ist eine eingetragene Marke oder Marke von VMware, Inc. in den USA und/oder anderen Ländern. Alle anderen in diesem Dokument erwähnten Bezeichnungen und Namen sind unter Umständen markenrechtlich geschützt.

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304, USA
www.vmware.com

Zweigniederlassung Deutschland
Freisinger Str. 3
85716 Unterschleißheim/Lohhof

Inhalt

Über dieses Handbuch 9

1 Installation von ThinApp 11
- ThinApp-Installationsanforderungen 11
  - Betriebssysteme, Anwendungen und Systeme, die von ThinApp unterstützt werden 11
  - Anwendungen, die von ThinApp nicht virtualisiert werden können 12
- Empfehlungen für die Installation von ThinApp 12
  - Verwenden eines sauberen Computers 12
  - Verwenden des frühesten Betriebssystems, das für die Benutzer erforderlich ist 13
- Installation von ThinApp 13
- Suchen nach ThinApp-Installationsdateien 13

2 Kapselung von Anwendungen 15
- Beschreibung des Kapselungsvorgangs 15
  - Bewertung der Anwendungsabhängigkeiten vor dem Kapselungsvorgang 15
  - Schließen von Anwendungen vor dem Kapselungsvorgang 16
- Kapseln einer Anwendung mit dem Setup Capture-Assistenten 16
  - Initiieren des Kapselungsvorgangs mit Vor- und Nachüberprüfungsabbildungen 16
  - Festlegen von Anwendungsverknüpfungen und Nachverfolgungsnamen 17
  - Festlegen von Benutzergruppen und Sandbox-Speicherorten 18
  - Festlegen des Dateisystemzugriffs 19
  - Festlegen der Einstellungen für die Anwendungs-Auslieferung 21
  - Erstellen von virtuellen Anwendungen 22
- Erweiterte Paketkonfiguration 23
  - Ändern der Einstellungen in der Package.ini-Datei 23
  - Ändern der Einstellungen in der ##Attributes.ini-Datei 24
  - Ändern der Isolationsmodi 24

3 Bereitstellen von Anwendungen 27
- ThinApp-Bereitstellungsoptionen 27
  - Bereitstellung von ThinApp mit Bereitstellungstools 27
  - Bereitstellen von ThinApp in der VMware View-Umgebung 27
  - Bereitstellen von ThinApp auf Netzwerkfreigaben 28
  - Bereitstellen von ThinApp unter Verwendung von ausführbaren Dateien 28
- Einrichten der Dateitypzuordnungen mit dem Dienstprogramm thinreg.exe 28
  - Auswirkung von Application Sync auf das Dienstprogramm thinreg.exe 28
  - Ausführen des Dienstprogramms thinreg.exe 29
  - Optionale thinreg.exe-Parameter 29
- Erstellen einer MSI-Datenbank 31
  - Anpassen von MSI-Dateien mit Package.ini-Parametern 31
  - Ändern der Datei Package.ini zum Erstellen von MSI-Dateien 32
- Steuern des Anwendungszugriffs mit Active Directory 33
  - Package.ini-Einträge für die Active Directory-Zugriffssteuerung 34
- Verwenden von ThinApp-Paketen mit Netzwerkstreaming 34
  - So funktioniert Anwendungsstreaming mit ThinApp 35
  - Anforderungen und Empfehlungen für Streamingpakete 36
  - Streamen von ThinApp-Paketen über das Netzwerk 37
- Verwenden von gekapselten Anwendungen mit anderen Systemkomponenten 37

Ausführen von Einfügevorgängen 37
Zugriff auf Drucker 37
Zugriff auf Treiber 37
Zugriff auf die lokale Festplatte, den Wechseldatenträger und die Netzwerkfreigaben 38
Zugriff auf die Systemregistrierung 38
Zugriff auf Netzwerk und Sockets 38
Verwenden von gemeinsam genutztem Arbeitsspeicher und Named Pipes 38
Verwenden von COM-, DCOM- und Out-of-Process-COM-Komponenten 38
Starten von Diensten 38
Verwenden von Dateitypzuordnungen 39
Beispielkonfiguration für Isolationsmodus je nach Bereitstellungskontext 39
Ansicht der Auswirkung von Isolationsmodus auf die Windows-Registrierung 39

4 Aktualisieren von Anwendungen 41
Anwendungs-Updates, die der Endbenutzer auslöst 41
Application Sync-Updates 41
Application Link-Updates 43
Anwendungs-Updates, die der Administrator auslöst 48
Erzwingen eines Updates von Application Sync auf Clientcomputern 48
Aktualisieren von Anwendungen mit Laufzeitänderungen 48
Automatische Anwendungs-Updates 50
Dynamische Updates ohne Administratorrechte 50
Aktualisieren von laufenden Anwendungen auf einer Netzwerkfreigabe 51
Dateisperren 51
Upgrade einer laufenden Anwendung 51
Sandbox-Überlegungen für aktualisierte Anwendungen 52

5 ThinApp – Überwachung und Problembehandlung 53
Bereitstellen von Informationen für den technischen Support 53
Protokoll-Monitor-Vorgänge 54
Problembehandlung mit Protokoll-Monitor 54
Ausführen von erweiterten Protokoll-Monitor-Vorgängen 55
Protokollformat 56
Problembehandlung bei bestimmten Anwendungen 61
Fehlerbehebung beim Setup der Registrierung für Microsoft Outlook 61
Anzeigen von Anhängen in Microsoft Outlook 61
Starten von Explorer.exe in der virtuellen Umgebung 62
Problembehandlung bei Versionskonflikten von Java Runtime Environment 63

A Konfigurieren von Paketparametern 65
Package.ini-Dateistruktur 66
Package.ini-Parameterplatzierung 66
Parameter, die auf Package.ini- oder ##Attributes.ini-Dateien angewendet werden 66
Konfigurieren der ThinApp-Laufzeit 66
NetRelaunch 66
RuntimeEULA 67
VirtualComputerName 67
Wow64 68
Konfigurieren des Dateisystems und der Registrierungsisolation 68
DirectoryIsolationMode 68
RegistryIsolationMode 69
Konfigurieren von Datei- und Protokollzuordnungen 69
FileTypes 69
Protocols 70
Konfigurieren der Build-Ausgabe 70

OutDir 70
ExcludePattern 70
Konfigurieren von Berechtigungen und Sicherheit 71
AccessDeniedMsg 71
AddPageExecutePermission 71
PermittedGroups 72
UACRequestedPrivilegesLevel 73
UACRequestedPrivilegesUIAccess 73
Konfigurieren von Objekten und DLL-Dateien 73
ExternalCOMObjects 73
ExternalDLLs 74
IsolatedMemoryObjects 74
IsolatedSynchronizationObjects 75
ObjectTypes 75
SandboxCOMObjects 76
VirtualizeExternalOutOfProcessCOM 76
Konfigurieren von Speicherplatz 77
CachePath 77
UpgradePath 77
VirtualDrives 78
Konfigurieren von Prozessen und Diensten 79
AllowExternalProcessModifications 79
AllowUnsupportedExternalChildProcesses 79
AutoShutdownServices 80
AutoStartServices 80
ChildProcessEnvironmentDefault 80
ChildProcessEnvironmentExceptions 81
Konfigurieren von Datei- und Blockgrößen 81
BlockSize 81
CompressionType 82
Konfigurieren von Symbolen 82
Icon 82
RetainAllIcons 83
Konfigurieren der Protokollierung 83
DisableTracing 84
LogPath 84
Konfigurieren von Versionen 84
CapturedUsingVersion 84
StripVersionInfo 84
Version.XXXX 85
Konfigurieren von lokalen Informationen 85
AnsiCodePage 85
LocaleIdentifier 85
LocaleName 86
Konfigurieren von einzelnen Anwendungen 86
CommandLine 86
Disabled 86
ReadOnlyData 87
ReserveExtraAddressSpace 87
Shortcut 88
Shortcuts 88
Source 88
WorkingDirectory 89
Konfigurieren von abhängigen Anwendungen mit Application Link 89
Pfadnamenformate für Application Link 90
RequiredAppLinks 90

OptionalAppLinks 91
Konfigurieren von Anwendungs-Updates mit Application Sync 92
AppSyncClearSandboxOnUpdate 92
AppSyncExpireMessage 92
AppSyncExpirePeriod 93
AppSyncURL 93
AppSyncUpdateFrequency 93
AppSyncUpdatedMessage 93
AppSyncWarningFrequency 94
AppSyncWarningMessage 94
AppSyncWarningPeriod 94
Konfigurieren von MSI-Dateien 94
MSIArpProductIcon 94
MSIDefaultInstallAllUsers 95
MSIFilename 95
MSIInstallDirectory 96
MSIManufacturer 96
MSIProductCode 96
MSIProductVersion 96
MSIRequireElevatedPrivileges 97
MSIUpgradeCode 97
MSIUseCabs 97
Konfigurieren von Sandbox-Speicher und Bestandsnamen 98
InventoryName 98
RemoveSandboxOnExit 99
SandboxName 99
SandboxNetworkDrives 99
SandboxPath 100
SandboxRemovableDisk 100

B ThinApp-Sandbox 101
Suchreihenfolge für die Sandbox 101
Steuern des Sandbox-Speicherorts 103
Speichern der Sandbox im Netzwerk 103
Speichern der Sandbox auf einem portablen Gerät 103
Sandbox-Struktur 104
Änderungen an der Sandbox 104
Auflisten virtueller Registrierungsinhalte mit vregtool 104

C Snapshot-Befehle und -Anpassungen 105
Methoden zur Verwendung des Dienstprogramms snapshot.exe 105
Erstellen von Snapshots des Computerstatus 105
Erstellen der Template Package.ini-Datei aus zwei Snapshot-Dateien 106
Erstellen des ThinApp-Projekts aus der Template Package.ini-Datei 106
Anzeigen von Inhalten einer Snapshot-Datei 107
Beispiele für snapshot.exe-Befehle 107
Erstellen eines Projekts ohne den Setup Capture-Assistenten 107
Anpassen der Snapshot.ini-Datei 108

D Virtuelles Dateisystem ThinApp 111
Virtuelle Dateisystemformate 111
Zusammengeführte und virtuelle Ansichten des Dateisystems 111
Verwenden von Ordnermakros 112
Liste der Ordnermakros 112
Verarbeitung von %SystemRoot% 113

E ThinApp-Skripts 115
Rückruffunktionen 115
Verwenden der Skripts in einer ThinApp-Umgebung 116
Beispiel: .bat 116
Beispiel: Timeout 117
Ändern der virtuellen Registrierung 117
Beispiel: .reg 117
Beispiel: Anhalten eines Dienstes 117
Beispiel: Kopieren einer Datei 117
Hinzufügen eines Wertes zur Systemregistrierung 118
API-Funktionen 119
AddForcedVirtualLoadPath 119
ExitProcess 120
ExpandPath 120
ExecuteExternalProcess 120
ExecuteVirtualProcess 121
GetBuildOption 121
GetFileVersionValue 121
GetCommandLine 122
GetCurrentProcessName 123
GetOSVersion 123
GetEnvironmentVariable 124
RemoveSandboxOnExit 124
SetEnvironmentVariable 124
SetfileSystemIsolation 125
SetRegistryIsolation 125
WaitForProcess 125

Glossar 127

Index 131

Über dieses Handbuch

Das *ThinApp-Benutzerhandbuch* enthält Informationen über die Installation von ThinApp sowie die Kapselung, die Bereitstellung und das Upgrade von Anwendungen. Sie können auf dieses Handbuch zurückgreifen, wenn Sie Parameter individuell anpassen und Scripting vornehmen möchten.

Zielgruppe

Dieses Handbuch ist ein Leitfaden für alle, die ThinApp installieren und gekapselte Anwendungen bereitstellen. Typische Anwender sind Systemadministratoren, die für Vertrieb und Wartung von Firmensoftwarepaketen verantwortlich sind.

Feedback zu diesem Dokument

VMware freut sich über Kommentare und Anregungen, um seine Dokumentation weiter zu verbessern. Senden Sie Ihr Feedback bitte an docfeedback@vmware.com.

Technischer Support und Schulungsressourcen

In den folgenden Abschnitten werden die verfügbaren technischen Supportressourcen beschrieben. Um auf die aktuelle Version dieses Handbuchs und anderer Bücher zuzugreifen, besuchen Sie bitte http://www.vmware.com/support/pubs.

Online- und Telefonsupport

Im Online-Support können Sie technische Unterstützung anfordern, Ihre Produkt- und Vertragsdaten abrufen und Produkte registrieren. Weitere Informationen finden Sie unter http://www.vmware.com/support.

Kunden mit entsprechenden Supportverträgen erhalten über den Telefonsupport die schnellste Hilfe bei Problemen der Prioritätsstufe 1. Weitere Informationen finden Sie unter http://www.vmware.com/support/phone_support.

Supportangebote

VMware stellt ein umfangreiches Supportangebot bereit, um Ihre geschäftlichen Anforderungen zu erfüllen. Weitere Informationen finden Sie unter http://www.vmware.com/support/services.

VMware Professional Services

VMware-Schulungskurse umfassen umfangreiche Praxisübungen, Fallbeispiele und Kursmaterialien, die zur Verwendung als Referenz-Tools bei der praktischen Arbeit vorgesehen sind. Kurse werden vor Ort, im Schulungsraum und live über das Internet angeboten. Für Pilotprogramme vor Ort und die Implementierung von Best Practices verfügt VMware Consulting Services über Angebote, um Ihnen bei Bewertung, Planung, Aufbau und Verwaltung Ihrer virtuellen Umgebung zu helfen. Um Informationen über Schulungskurse, Zertifizierungsprogramme und Consultingdienste zu erhalten, besuchen Sie bitte http://www.vmware.com/services.

Installation von ThinApp 1

Die ThinApp-Software isoliert und speichert Anwendungen, vereinfacht Anwendungsanpassungen sowie die Bereitstellung von Anwendungen auf verschiedenen Betriebssystemen und eliminiert Anwendungskonflikte.

Dieser Abschnitt umfasst die folgenden Themen:

- „ThinApp-Installationsanforderungen“ auf Seite 11
- „Empfehlungen für die Installation von ThinApp“ auf Seite 12
- „Installation von ThinApp“ auf Seite 13
- „Suchen nach ThinApp-Installationsdateien“ auf Seite 13

ThinApp-Installationsanforderungen

Überprüfen Sie vor der Installation von ThinApp die Anforderungen an die Betriebssysteme und die gekapselten Anwendungen.

Betriebssysteme, Anwendungen und Systeme, die von ThinApp unterstützt werden

ThinApp unterstützt folgende Betriebssysteme, Anwendungen und Systeme:

- 32-Bit-Plattformen, einschließlich Windows NT, Windows 2000, Windows XP, Windows XPE, Windows 2003 Server, Windows Vista, Windows Server 2008
- 64-Bit-Plattformen, einschließlich Windows XP 64 Bit, Windows 2003 64 Bit, Windows Vista 64 Bit, Windows Server 2008 64 Bit
- Auf 32-Bit-Windows-Betriebssystemen ausgeführte 16-Bit-Anwendungen
- Auf 32-Bit- und 64-Bit-Windows-Betriebssystemen ausgeführte 32-Bit-Anwendungen
- Terminalserver und Citrix Xenapp

ThinApp unterstützt japanische Anwendungen, die auf japanischen Betriebssystemen gekapselt und ausgeführt werden.

Folgende Betriebssysteme und Anwendungen werden von ThinApp nicht unterstützt:

- 16-Bit- oder Nicht-x86-Plattformen wie Windows CE
- Auf 32-Bit- oder 64-Bit-Windows-Betriebssystemen ausgeführte 64-Bit-Anwendungen
- Auf 64-Bit-Windows-Betriebssystemen ausgeführte 16-Bit-Anwendungen

Anwendungen, die von ThinApp nicht virtualisiert werden können

ThinApp kann einige Anwendungen nicht in virtuelle Anwendungen konvertieren und blockiert möglicherweise bestimmte Anwendungsfunktionen.

Zur Bereitstellung der folgenden Anwendungstypen sind herkömmliche Installationstechniken erforderlich:

- Anwendungen, die die Installation von Gerätetreibern im Kernelmodus erfordern.

  ODBC-Treiber funktionieren, da sie Treiber im Benutzermodus sind.
- Antivirus und persönliche Firewalls
- Scannertreiber und Druckertreiber
- Einige VPN-Clients

Gerätetreiber

Anwendungen, für die Gerätetreiber erforderlich sind, funktionieren nach der Paketierung durch ThinApp nicht. Diese Gerätetreiber müssen in ihrem ursprünglichen Format auf dem Hostcomputer installiert werden. Da ThinApp keine virtualisierten Gerätetreiber unterstützt, können Sie ThinApp nicht verwenden, um Antivirus, VPN-Clients, persönliche Firewalls und Dienstprogramme für die Festplatte und zum Bereitstellen von Volumes zu virtualisieren.

Wird Adobe Acrobat gekapselt, können PDF-Dateien geöffnet, bearbeitet und gespeichert werden, doch der PDF-Druckertreiber, der das Speichern von Dokumenten im PDF-Format ermöglicht, kann nicht angezeigt oder verwendet werden.

Shell-Integration

Einige Anwendungen, die Shell-Integration bieten, erfahren möglicherweise Funktionsbeschränkungen, wenn sie in einem ThinApp-Paket vorhanden sind. So kann beispielsweise eine virtuelle Anwendung, die in Windows Explorer integriert ist, keine spezifischen Einträge zu den Windows Explorer-Kontextmenüs hinzufügen.

Über das Netzwerk verfügbare DCOM-Dienste

ThinApp isoliert COM- und DCOM-Dienste. Auf Anwendungen, die DCOM-Dienste installieren, kann auf dem lokalen Computer nur durch andere gekapselte Anwendungen zugegriffen werden, die in derselben ThinApp-Sandbox ausgeführt werden. ThinApp unterstützt virtuelles DCOM und COM auf demselben Computer; Netzwerk-DCOM wird jedoch nicht unterstützt.

Globale Hook-DLLs

Einige Anwendungen verwenden die API-Funktion `SetWindowsHookEx`, um sämtlichen Prozessen auf dem Hostcomputer eine DLL hinzuzufügen. Die hinzugefügte DLL fängt Windows-Meldungen ab, um Tastatur- und Mauseingaben von anderen Anwendungen zu kapseln. ThinApp ignoriert Anforderungen von Anwendungen, die die Funktion `SetWindowsHookEx` für den Versuch einsetzen, globale Hook-DLLs zu installieren. ThinApp beschränkt möglicherweise die Funktionen der Anwendung.

Empfehlungen für die Installation von ThinApp

Berücksichtigen Sie bei der Installation von ThinApp die Empfehlungen und die bewährten Methoden.

Verwenden eines sauberen Computers

VMware empfiehlt, für die Installation von ThinApp sauberen Computer zu verwenden, da die Umgebung sich auf den Kapselungsvorgang der Anwendung auswirkt. Ein sauberer Computer ist eine physische oder virtuelle Maschine, auf der nur ein Windows-Betriebssystem installiert ist. In einer Unternehmensumgebung mit einem Basis-Desktop-Bild, ist das Basis-Desktop-Bild ein sauberer Computer. Möglicherweise wurden auf dem Desktop-Computer bereits einige Komponenten und Bibliotheken vorinstalliert.

Installationsassistenten von Anwendungen überspringen Dateien, die bereits auf dem Computer vorhanden sind. Überspringt der Installationsassistent Dateien, schließt das ThinApp-Paket sie nicht während der Kapselung der Anwendung ein. Die Anwendung kann möglicherweise nicht auf anderen Computern, auf denen die Dateien nicht vorhanden sind, ausgeführt werden. Ein sauberer Computer ermöglicht während der Kapselung die schnelle Überprüfung des Dateisystems und der Registrierung des Computers.

Wenn Sie ThinApp installieren und eine Anwendung auf einem Computer kapseln, auf dem Microsoft .NET 2.0 bereits installiert ist, wird .NET 2.0 nicht in das ThinApp-Paket eingeschlossen. Die gekapselte Anwendung wird nur auf Computern ausgeführt, auf denen .NET 2.0 bereits installiert ist.

Verwenden virtueller Maschinen als sauberer Computer

Die einfachste Weise, einen sauberen Computer einzurichten, ist das Erstellen einer virtuellen Maschine. Sie können Windows auf der virtuellen Maschine installieren und einen Snapshot der gesamten virtuellen Maschine im sauberen Zustand erstellen. Nach der Kapselung einer Anwendung können Sie den Snapshot wiederherstellen und die virtuelle Maschine in den sauberen Zustand zurücksetzen, die bereit für die nächste Kapselung der Anwendung ist.

Zum Erstellen virtueller Maschinen können Sie VMware Workstation oder andere VMware-Produkte verwenden. Informationen über VMware-Produkte finden Sie auf der Website von VMware.

Verwenden des frühesten Betriebssystems, das für die Benutzer erforderlich ist

Installieren Sie ThinApp mit der frühesten Version des Betriebssystems, das unterstützt werden soll, auf einer neu aufgesetzten Machine. In den meisten Fällen ist die früheste Plattform Windows 2000 oder Windows XP. Einige unter Windows XP gekapselte Pakete funktionieren unter Windows 2000. In einigen Fällen enthält Windows XP einige DLLs, die Windows 2000 nicht enthält. ThinApp schließt diese DLLs aus dem gekapselten Anwendungspaket aus, wenn die Anwendung diese DLLs typischerweise installiert.

Nachdem Sie ein ThinApp-Anwendungpaket erstellt haben, können Sie Dateien im Paket mit aktualisierten Versionen überschreiben und die Anwendung ohne den Kapselungsvorgang neu erstellen.

Installation von ThinApp

Verwenden der ausführbaren ThinApp-Datei zum Installieren von ThinApp.

So installieren Sie die ThinApp-Software:

1. Laden Sie ThinApp auf eine physische oder virtuelle neu aufgesetzte Machine mit Windows-System herunter.
2. Doppelklicken Sie auf die ausführbare ThinApp-Datei.
3. Klicken Sie im Dialogfeld **Patentlisten (Patent Lists)** auf **Weiter (Next)**.
4. Akzeptieren Sie die Lizenzvereinbarung, geben Sie die Seriennummer und einen Lizenzanzeigenamen ein. Dieser wird beim Starten von Anwendungen angezeigt wird, die ThinApp kapselt.
5. Klicken Sie auf **Installieren (Install)**.

   ThinApp wird installiert.

Suchen nach ThinApp-Installationsdateien

Bei der Installation von ThinApp wird unter `C:\Programme\VMware` das `VMware ThinApp`-Verzeichnis generiert. Möglicherweise müssen Sie zur Anzeige von Lizenzinformationen oder zur Durchführung bestimmter Vorgänge, wie beispielsweise dem Starten des Dienstprogramms Protokoll-Monitor zur Ansicht der letzten Aktivitäten, nach Dateien in diesem Verzeichnis suchen.

Die folgenden wichtigen Dateien im `VMware ThinApp`-Verzeichnis haben Auswirkungen auf ThinApp-Vorgänge:

- **AppSync.exe** – Hält gekapselte Anwendungen auf dem Stand der neuesten verfügbaren Version.

- **logging.dll** – Generiert `.trace`-Dateien.
- **dll_dump.exe** – Listet alle gekapselten Anwendungen, die aktuell auf einem System ausgeführt werden.
- **log_monitor.exe** – Zeigt den Ausführungsverlauf und die Fehler einer Anwendung an.
- **sbmerge.exe** – Verbindet in der Anwendungs-Sandbox eingetragene Laufzeitänderungen mit dem ThinApp-Projekt und aktualisiert die gekapselte Anwendung.
- **Setup Capture.exe** – Kapselt Anwendungen und konfiguriert sie mithilfe eines Assistenten.
- **snapshot.exe** – Vergleicht während der Kapselung der Anwendung die Umgebung vor und nach der Installation.

  ThinApp startet dieses Dienstprogramm während des Setup Capture-Prozesses.
- **snapshot.ini** – Speichert Einträge für die virtuelle Registrierung und das virtuelle Dateisystem, die ThinApp während des Kapselungsvorgangs einer Anwendung ignoriert.

  Die `snapshot.exe`-Datei verweist auf die `snapshot.ini`-Datei. Fortgeschrittene Benutzer können die `snapshot.ini`-Datei modifizieren, um sicherzustellen, dass ThinApp bestimmte Einträge bei der Erstellung eines Anwendungspakets nicht kapselt.
- **template.msi** – Erstellt die MSI-Dateien.

  Diese Vorlage kann angepasst werden, um sicherzustellen, dass die von ThinApp generierten `.msi`-Dateien den Bereitstellungsverfahren und -normen des Unternehmens gerecht werden. Zum Beispiel können Registrierungseinstellungen hinzufügt werden, die ThinApp als Bestandteil der Installation zu einem Clientcomputer hinzufügen soll.
- **thinreg.exe** – Registriert gekapselte Anwendungen auf einem Computer.

  Diese Registrierung beinhaltet das Einrichten von Verknüpfungen und des Start-Menüs und das Festlegen von Dateitypzuordnungen, die den Start der Anwendungen ermöglichen.
- **tlink.exe** – Verbindet Schlüsselmodule während des Build-Prozesses der gekapselten Anwendung.
- **vftool.exe** – Kompiliert das virtuelle Dateisystem während des Build-Prozesses der gekapselten Anwendung.
- **vregtool.exe** – Kompiliert die virtuelle Registrierung während des Build-Prozesses der gekapselten Anwendung.

Kapselung von Anwendungen 2

Sie können Anwendungen mit dem Setup Capture-Assistenten kapseln. Der Kapselungsvorgang verpackt eine Anwendung in eine virtuelle Umgebung und legt die anfänglichen Anwendungsparameter fest. Informationen zur Kapselung von Anwendungen über die Befehlszeile erhalten Sie unter Anhang C, „Snapshot-Befehle und -Anpassungen“, auf Seite 105.

In den nachstehenden Erläuterungen dient Mozilla Firefox als Schlüsselbeispiel für die Kapselung von Anwendungen. Folgende Themen werden behandelt:

- „Beschreibung des Kapselungsvorgangs“ auf Seite 15
- „Kapseln einer Anwendung mit dem Setup Capture-Assistenten“ auf Seite 16
- „Erweiterte Paketkonfiguration“ auf Seite 23

Beschreibung des Kapselungsvorgangs

Der Kapselungsvorgang umfasst folgende Phasen:

- Überprüfen einer Baselineabbildung der neu aufgesetzten Maschine.
- Installation der Anwendung, die von ThinApp gekapselt werden soll.
- Konfiguration der spezifischen Einstellungen für die Anwendung.

  Sie können verschiedene Einstellungen festlegen, beispielsweise Firefox als Standardbrowser, eine Startseite und Standardsicherheitseinstellungen.
- Überprüfen des Computers nach der Anwendungsinstallation.

  ThinApp bewertet die Unterschiede zwischen der ursprünglichen Baselineabbildung und dieser Abbildung.
- Konfiguration der ThinApp-Parameter, um Bereiche wie die Komprimierung ausführbarer Dateien, den Speicherort für die Sandbox und den Zugriff der Domänen-Benutzer auf Anwendungen anzupassen.
- Erstellen des virtuellen Anwendungspakets zur Verteilung.

Bewertung der Anwendungsabhängigkeiten vor dem Kapselungsvorgang

Bevor Sie eine Anwendung kapseln, müssen Sie bewerten, ob für die Anwendung Abhängigkeiten zu anderen Anwendungen, Bibliotheken oder Umgebungen bestehen und ob diese Abhängigkeiten gekapselt werden sollen. VMware empfiehlt, das Dienstprogramm Application Link zu verwenden, um separate Komponenten während der Laufzeit zu verknüpfen. Siehe Kapitel 4, „Aktualisieren von Anwendungen“, auf Seite 41.

Schließen von Anwendungen vor dem Kapselungsvorgang

Um das Dateisystem zu schützen, empfiehlt VMware das Schließen von Anwendungen wie beispielsweise Anti-Virenprogrammen, die während des Erstellens von Snapshots durch ThinApp das Dateisystem verändern könnten.

Kapseln einer Anwendung mit dem Setup Capture-Assistenten

Beim Setup Capture-Prozess wird eine Anwendung paketiert und die Eingangsparameter der Anwendung werden festgelegt. Wenn Sie eine virtuelle Maschine verwenden, empfiehlt es sich, vor dem Ausführen des Assistenten einen Snapshot zu erstellen. Ein Snapshot des ursprünglichen sauberen Zustands ermöglicht Ihnen das Zurücksetzen auf diesen Snapshot, wenn Sie eine weitere Anwendung kapseln möchten.

Initiieren des Kapselungsvorgangs mit Vor- und Nachüberprüfungsabbildungen

Der Setup Capture-Assistent startet den Kapselungsvorgang durch die Erstellung mehrerer Systemüberprüfungen zur Bewertung der Umgebung vor und nach der Anwendungsinstallation.

So starten Sie den Kapselungsvorgang mit Vor- und Nachüberprüfungsabbildungen

1 Laden Sie die zu kapselnden Anwendungen herunter.

Laden Sie beispielsweise `Firefox Setup 2.0.0.3.exe` herunter und kopieren Sie das Programm auf die neu aufgesetzte Machine, mit der Sie arbeiten.

2 Wählen Sie auf dem Desktop **Start** > **Alle Programme** > **VMware** > **ThinApp Setup Capture**.

3 (Optional) Klicken Sie im Dialogfeld **System vorüberprüfen (Prescan System)** auf **Erweiterte Speicherorte zur Überprüfung (Advanced scan locations)**, um die Laufwerke und Registrierungsstrukturen zum Überprüfen auszuwählen.

Es empfiehlt sich, einen anderen Speicherort als das Laufwerk `C:\` zu überprüfen, wenn Sie Anwendungen auf ein anderes Laufwerk installieren. In seltenen Fällen, wenn Sie wissen, dass das Installationsprogramm der Anwendung die Registrierung nicht verändert, empfiehlt es sich, die Registrierungsstruktur nicht zu überprüfen.

4 Klicken Sie auf **Überprüfen (Scan)**, um eine Baselineabbildung der Festplatte und der Registrierungsdateien zu erstellen.

Mit Windows XP dauert die Überprüfung ungefähr 10 Sekunden.

5 Wenn die Seite **Anwendung installieren (Install Application)** angezeigt wird, minimieren Sie den Setup Capture-Assistenten und installieren Sie die Anwendungen, die gekapselt werden sollen.

Doppelklicken Sie zum Beispiel auf `Firefox Setup 2.0.0.3.exe`, um Firefox zu installieren. Starten Sie das System neu, wenn die Anwendung nach der Installation neu gestartet werden muss. Beim Neustart wird der Setup Capture-Assistent erneut gestartet.

6 Nehmen Sie erforderliche Konfigurationsänderungen vor, um die Richtlinien Ihres Unternehmens einzuhalten, beispielsweise die Festlegung spezieller Sicherheitseinstellungen oder einer bestimmten Startseite.

Wenn Sie zu diesem Zeitpunkt keine Konfigurationsänderungen vornehmen, müssen die einzelnen Benutzer Anpassungen vornehmen.

7 (Optional) Starten Sie die Anwendung und reagieren Sie auf alle Eingabeaufforderungen, bevor Sie mit dem Setup Capture-Assistenten fortfahren.

Wenn Sie zu diesem Zeitpunkt nicht auf Eingabeaufforderungen reagieren, müssen die einzelnen Benutzer der Anwendung dies beim ersten Starten tun.

8 Schließen Sie die Anwendung.

9 Maximieren Sie den Setup Capture-Assistenten, klicken Sie auf **Nachüberprüfung (Postscan)**, um mit einer weiteren Überprüfung des Computers fortzufahren und klicken Sie dann auf **OK**, um den Nachüberprüfungsvorgang zu bestätigen.

ThinApp speichert die Unterschiede zwischen der ersten Baselineabbildung und dieser Abbildung in einem virtuellen Dateisystem und einer virtuellen Registrierung.

Festlegen von Anwendungsverknüpfungen und Nachverfolgungsnamen

Einstiegspunkte sind die ausführbaren Dateien, die die virtuelle Anwendung starten und den Zugriff darauf ermöglichen. Welche Einstiegspunkte zur Auswahl stehen, hängt von den ausführbaren Dateien ab, die Ihre gekapselte Anwendung bei der Installation generiert. Wenn Sie zum Beispiel Microsoft Office installieren, können Sie Einstiegspunkte wählen für Microsoft Word, Microsoft Excel und weitere Anwendungen, die während einer Microsoft Office-Installation installiert werden. Wenn Sie Firefox installieren, empfiehlt es sich, `Mozilla Firefox.exe` und `Mozilla Firefox (SafeMode).exe` zu wählen, falls Benutzer den Zugang im abgesicherten Modus benötigen.

Während des Build-Prozesses am Schluss des Setup Capture-Assistenten generiert ThinApp für jeden ausgewählten Einstiegspunkt eine ausführbare Datei. Wenn Sie die Anwendung als MSI-Datei bereitstellen oder das Dienstprogramm `thinreg.exe` verwenden, verweisen die auf den Endbenutzer-Desktops erstellten Verknüpfungen von Desktop und **Start**-Menü auf diese Einstiegspunkte.

ThinApp bietet die folgenden Einstiegspunkte zur Problembehandlung oder zum Debuggen Ihrer Umgebung:

- `cmd.exe` – Gibt eine Eingabeaufforderung in einem virtuellen Kontext aus, mit dem Sie das virtuelle Dateisystem anzeigen können.
- `regedit.exe` – Startet den Registrierungseditor in einem virtuellen Kontext, mit dem Sie die virtuelle Registrierung anzeigen können.
- `iexplore.exe` – Startet `iexplore.exe` in einem virtuellen Kontext, damit Sie virtualisierte ActiveX -Steuerungen testen können.

Einstiegspunkte starten native ausführbare Dateien in einem virtuellen Kontext. Einstiegspunkte erzeugen keine virtuellen Pakete von `cmd.exe`, `regedit.exe` oder `iexplore.exe`.

Wenn Sie nicht vorhersehen können, ob Debugging oder eine Problembehandlung der Umgebung erforderlich wird, können Sie stattdessen den Parameter `Disabled` in der `Package.ini`-Datei verwenden, um diese Einstiegspunkte zu einem späteren Zeitpunkt zu aktivieren. Siehe „Disabled“ auf Seite 86.

Im Gegensatz zu Einstiegspunkten ist der primäre Datencontainer die einzige Datei, die das schreibgeschützte virtuelle Dateisystem und die virtuelle Registrierung enthält. Sie können die primäre Datencontainerdatei bestimmen, indem Sie einen Einstiegspunkt wählen oder einen Namen für den Container eingeben. Dieser Name erscheint in der `Package.ini`-Datei, gefolgt von einer `ReadOnlyData`-Parameterzeile. Siehe „ReadOnlyData“ auf Seite 87.

Der Bestandsname vereinfacht die interne Nachverfolgung der Anwendung in der `Package.ini`-Datei.

So legen Sie Anwendungsverknüpfungen und Nachverfolgungsnamen im Setup Capture-Assistenten fest

1 Aktivieren Sie auf der Seite **Verknüpfungen für Anwendungszugriff auswählen (Select Application Access Shortcuts)** die Kontrollkästchen für Einstiegspunkte, auf die Benutzer Zugriff haben.

Der Assistent aktualisiert die Liste mit ausführbaren Dateien, die ThinApp während des Kapselungsvorgangs installiert hat, und wählt automatisch die ausführbaren Dateien aus, auf die direkt über Verknüpfungen auf dem Desktop oder im **Start**-Menü zugegriffen werden konnte.

2 (Optional) Wenn Sie Ihre Umgebung debuggen möchten, aktivieren Sie das Kontrollkästchen **Einstiegspunkte für Debugging anzeigen (Show entry points used for debugging)**, um die Einstiegspunkte `iexplore.exe`, `regedit.exe` und `cmd.exe` aus der Liste auszuwählen.

3 Wählen Sie aus der anhand der ausgewählten Einstiegspunkte erstellten Liste den primären Datencontainer: die Datei, in der die virtuellen Dateien und Registrierungsdaten gespeichert werden.

   - Umfasst der primäre Datencontainer weniger als 200 MB, erstellt ThinApp eine `.exe`-Datei als primären Container. Für eine kleine Anwendung wie Firefox kann jede `.exe`-Datei als Hauptdatencontainer dienen.
   - Ist der primäre Datencontainer größer als 200 MB, erstellt ThinApp eine eigene `.dat`-Datei als primären Container, da Windows XP und Windows 2000 nicht über Verknüpfungssymbole für große `.exe`-Dateien verfügen. Das Problem lässt sich durch Generieren getrennter kleiner `.exe`-Dateien zusammen mit der `.dat`-Datei lösen.
   - Liegt die Größe der primären Containerdatei zwischen 200 MB und 1,5 GB, erstellt ThinApp die standardmäßige `.dat`-Datei, ausgenommen Sie wählen eine `.exe`-Datei, die die standardmäßige `.dat`-Datei außer Kraft setzt.

4 Wenn Sie eine `.exe`-Datei zum Außerkraftsetzen der standardmäßigen `.dat`-Datei bei einer primären Containergröße von 200 MB bis 1,5 GB wählen, ignorieren Sie den angezeigten Warnhinweis.

   Die Auswahl einer `.exe`-Datei ermöglicht es, alle Anwendungen ordnungsgemäß auszuführen; die korrekte Anzeige der Symbole könnte hierdurch jedoch verhindert werden.

5 Wenn Sie keinen primären Datencontainer auswählen können, geben Sie einen primären Datencontainernamen ein.

   Wenn die Verwendung des Dienstprogramms Application Sync geplant ist, um eine gekapselte Anwendung zu aktualisieren, verwendet ThinApp während des Vorgangs den primären Datencontainernamen. Sie müssen für mehrere Versionen der Anwendung denselben Namen verwenden. Möglicherweise können Sie denselben primären Datencontainernamen nicht aus der Liste wählen. Zum Beispiel haben Microsoft Office 2003 und Microsoft Office 2007 keine gemeinsamen Namen für Einstiegspunkte.

6 (Optional) Ändern Sie den Bestandsnamen, den ThinApp für die interne Nachverfolgung der Anwendung in der `Package.ini`-Datei verwendet.

   Durch die Verwendung des Dienstprogrammes `thinreg.exe` oder die Bereitstellung der gekapselten Anwendung als MSI-Datei wird der Bestandsname im Dialogfeld „Software“ von Windows angezeigt.

Festlegen von Benutzergruppen und Sandbox-Speicherorten

ThinApp kann mithilfe von Active Directory-Gruppen den Zugriff auf die Anwendung und den Speicherort der Sandbox autorisieren. Zum Beispiel können Sie den Zugriff auf eine Anwendung einschränken, um sicherzustellen, dass Benutzer diese nicht an unbefugte Benutzer übergeben. ThinApp unterstützt keine verschachtelten Active Directory-Gruppen. Wenn zum Beispiel ein Benutzer zur Gruppe A gehört und Gruppe A zur Gruppe B gehört, kann ThinApp den Benutzer nur als Mitglied der Gruppe A erkennen und nicht als Mitglied von Gruppe A und Gruppe B.

Die Sandbox ist das Verzeichnis, in dem alle Änderungen, die eine gekapselte Anwendung vornimmt, gespeichert werden. Sobald Sie die Anwendung das nächste Mal starten, werden diese Änderungen aus der Sandbox integriert. Wenn Sie das Sandbox-Verzeichnis löschen, wird die Anwendung auf den gekapselten Status zurückgesetzt. Wenn eine Anwendung ein Problem hat und Sie sie wieder in den Originalzustand zurückversetzen möchten, sollten Sie die Sandbox löschen. Weitere Informationen über die Sandbox erhalten Sie in Anhang B, „ThinApp-Sandbox“, auf Seite 101.

So legen Sie Benutzergruppen und Sandbox-Speicherorte im Setup Capture-Assistenten fest

1 (Optional) Klicken Sie auf der Seite **Benutzergruppen und Sandbox-Speicherorte einrichten (Set Up User Groups and Sandbox Location)** und klicken Sie auf **Hinzufügen (Add)**, um Active Directory-Informationen festzulegen.

Option	Aktion
Objekttypen (Object Types)	Gibt Objekte an.
Speicherorte (Locations)	Gibt einen Speicherort in der Gesamtstruktur an.
Objektnamen (Object names) (manuell eintragen)	Sucht nach Objektnamen.
Erweitert (Advanced)	Sucht Benutzernamen in der Active Directory-Gesamtstruktur.
Häufig gestellte Fragen (Common Queries) (unter **Erweitert (Advanced)**)	Sucht nach Gruppen entsprechend den Namen, Beschreibungen, deaktivierten Konten, Passwörtern und Tagen seit der letzten Anmeldung.

2 Wählen Sie den Speicherort der ThinApp-Sandbox.

Sie können sie auf einem lokalen Computer bereitstellen, auf einem USB-Stick mitnehmen oder in einem Netzwerkpfad speichern.

Wenn Sie die Sandbox auf einem lokalen Computer bereitstellen, verwenden Sie das Profil des Benutzers. Wenn Sie die Sandbox auf einem Netzlaufwerk speichern, geben Sie den absoluten Pfad zum Speicherort ein, an dem Sie die Sandbox erstellen möchten. Ein Beispiel für einen Pfad ist `\\thinapp\sandbox\Firefox`. Sie können selbst dann einen Netzwerkspeicherort wählen, wenn eine Anwendung auf einem lokalen Computer installiert ist.

Festlegen des Dateisystemzugriffs

Isolationsmodi helfen bei der Festlegung von Änderungen des virtuellen und des physischen Umfeldes. Während des Kapselungsvorgangs können Sie die Isolationsmodi Merged und WriteCopy einrichten, um unterschiedliche Schreibzugriffsebenen für das physische Dateisystem zu bestimmen. Der Assistent stellt den Isolationsmodus „Full (Voll)“ nicht zur Verfügung. Informationen zum Isolationsmodus „Full (Voll)“, der unabhängig vom Assistenten verfügbar ist, erhalten Sie unter „Ändern der Isolationsmodi“ auf Seite 24.

Der Schlüsseleffekt bei der Auswahl der Isolationsmodi „Merged (Zusammengeführt)“ und „WriteCopy“ im Setup Capture-Assistenten liegt im Wert des `DirectoryIsolationMode`-Parameters in der `Package.ini`-Datei. Dieser Parameter steuert den Standard-Isolationsmodus für beim Kapselungsvorgang erstellte Dateien, sofern nicht ein anderer Isolationsmodus für ein einzelnes Verzeichnis in der `##Attributes.ini`-Datei vorhanden ist. Informationen über den Parameter `DirectoryIsolationMode` erhalten Sie unter „DirectoryIsolationMode“ auf Seite 68.

Mit dem Isolationsmodus „Merged (Zusammengeführt)“ kann die Anwendung Elemente auf dem physischen Dateisystem außerhalb des virtuellen Anwendungspakets ändern. Einige Anwendungen sind von DLLs und Registrierungsdaten im lokalen Systemabbild abhängig. Der Vorteil des Modus „Merged (Zusammengeführt)“ liegt darin, dass von Benutzern gespeicherte Dokumente im physischen System auf dem vom Benutzer erwarteten Speicherort anstatt in der Sandbox abgelegt werden. Der Nachteil ist, dass dieser Modus das Systemabbild überhäufen könnte. Ein Beispiel für den Rückstand können Markierungen für die Erstausführung von Shareware-Anwendungen auf zufälligen Computerspeicherorten als Teil des Lizenzierungsvorgangs sein.

Wenn Sie im Setup Capture-Assistenten den Isolationsmodus „Merged (Zusammengeführt)“ auswählen, führt ThinApp folgende Vorgänge durch:

- ThinApp setzt den `DirectoryIsolationMode`-Parameter in der `Package.ini`-Datei auf `Merged`.
- ThinApp weist den Isolationsmodus „Merged (Zusammengeführt)“ den folgenden Verzeichnissen zu:

- %Personal%
- %Desktop%
- %SystemSystem%\spool

Wenn Sie Dokumente auf dem Desktop und im Ordner `Eigene Dateien` speichern, speichert ThinApp die Dokumente auf dem physischen System, unabhängig vom Modus „Merged (Zusammengeführt)“, weil „Merged (Zusammengeführt)“ sich auf die an globalen Speicherorten wie `C:\Eigene Dateien` gespeicherten Dokumente auswirkt.

- ThinApp schließt einige Speicherorte vom Isolationsmodus „Merged“ aus und weist den Isolationsmodus „WriteCopy“ den folgenden Verzeichnissen und ihren Unterverzeichnissen zu:
  - `%AppData%`
  - `%Common AppData%`
  - `%Local AppData%`
  - `%Program Files Common%`
  - `%ProgramFilesDir%`
  - `%SystemRoot%`
  - `%SystemSystem%`
- ThinApp weist den Isolationsmodus „Full (Voll)“ jedem neuen Verzeichnis zu, das die Anwendung während der Installation erstellt.

Die Option **„**Merged (Zusammengeführt)**“** im Setup Capture-Assistenten hat dieselbe Auswirkung wie die Einstellung „Merged“ in der `Package.ini`-Datei, aber die Ausnahmen im Verzeichnis, die den Isolationsmodus „WriteCopy“ verwenden, gelten nur für die Option im Assistenten. Der Assistent konfiguriert die Verzeichnisausnahmen für Sie und fügt innerhalb der Verzeichnisse `##Attributes.ini`-Dateien hinzu. Um dasselbe Ergebnis außerhalb des Assistenten zu erzielen, müssen Sie diese Verzeichnisausnahmen manuell konfigurieren.

Mit dem Isolationsmodus „WriteCopy“ kann ThinApp Schreibvorgänge abfangen und in die Sandbox umleiten. VMware empfiehlt den Modus „WriteCopy“ für ältere oder nicht vertrauenswürdige Anwendungen. Obwohl das Auffinden von Benutzerdatendateien, die sich in der Sandbox statt im aktuellen System befinden, durch diesen Modus möglicherweise erschwert wird, ist dieser Modus für gesperrte Desktops nützlich, wenn Sie Benutzer daran hindern möchten, das Betriebsdateisystem und Registrierungsdateien zu beeinflussen.

Wenn Sie im Setup Capture-Assistenten den Isolationsmodus „WriteCopy“ auswählen, führt ThinApp folgende Vorgänge durch:

- ThinApp setzt den `DirectoryIsolationMode`-Parameter in der `Package.ini`-Datei auf `WriteCopy`.
- ThinApp weist den Isolationsmodus „WriteCopy“ folgenden Verzeichnissen zu:
  - `%AppData%`
  - `%Common AppData%`
  - `%Local AppData%`
  - `%Program Files Common%`
  - `%ProgramFilesDir%`
  - `%SystemRoot%`
  - `%SystemSystem%`
- ThinApp weist den Isolationsmodus „Merged“ den folgenden Verzeichnissen zu:
  - %Personal%
  - %Desktop%

- %SystemSystem%\spool

- ThinApp weist den Isolationsmodus „Full (Voll)“ jedem neuen Verzeichnis zu, das die Anwendung während der Installation erstellt.

Die Option **WriteCopy** im Setup Capture-Assistenten hat dieselbe Auswirkung wie die Einstellung „WriteCopy“ in der `Package.ini`-Datei, aber die Ausnahmen im Verzeichnis, die den Isolationsmodus „WriteCopy“ verwenden, gelten nur für die Option im Assistenten. Der Assistent konfiguriert die Verzeichnisausnahmen für Sie und fügt innerhalb der Verzeichnisse `##Attributes.ini`-Dateien hinzu. Um dasselbe Ergebnis außerhalb des Assistenten zu erzielen, müssen Sie diese Verzeichnisausnahmen manuell konfigurieren.

Unabhängig vom ausgewählten Isolationsmodus behandelt ThinApp Schreibvorgänge auf Netzlaufwerke gemäß dem `SandboxNetworkDrives`-Parameter in der `Package.ini`-Datei. Dieser Parameter hat einen Standardwert von 0, der Schreibvorgänge zum physischen Laufwerk lenkt. ThinApp behandelt Schreibvorgänge auf Wechseldatenträger gemäß dem `SandboxRemovableDrives`-Parameter in der `Package.ini`-Datei. Dieser Parameter hat einen Standardwert von 0, der Schreibvorgänge zum physischen Laufwerk lenkt.

Alle Laufzeitänderungen für virtuelle Elemente in der gekapselten Anwendung sind in der Sandbox gespeichert, unabhängig von der Einstellung des Isolationsmodus. Während der Laufzeit sind virtuelle und physische Registrierungselemente für eine Anwendung nicht unterscheidbar, aber virtuelle Registrierungselemente haben immer Vorrang vor physischen, wenn beide am selben Speicherort vorhanden sind. Wenn virtuelle und physische Einträge am selben Speicherort vorhanden sind, beeinflussen Isolationsmodi den Zugriff auf diese Einträge nicht, weil die Anwendung immer mit virtuellen Elementen interagiert. Wenn Updates externer Gruppenrichtlinien getrennt vom Paket durch die physische Registrierung auftreten, müssen Sie möglicherweise virtuelle Registrierungselemente aus einem Paket entfernen und bestätigen, dass das übergeordnete Element dieser virtuellen Registrierungselemente nicht den Isolationsmodus „Full (Voll)“ verwendet. Da untergeordnete Elemente Isolationsmodi von übergeordneten Elementen übernehmen, kann der Isolationsmodus „Full (Voll)“ in einem übergeordneten Element die Sichtbarkeit von physischen untergeordneten Elementen für eine Anwendung blockieren.

So legen Sie den Dateisystemzugriff im Setup Capture-Assistenten fest

Wählen Sie auf der Seite **Festlegen des Dateisystemzugriffs (Specify File System Access)** den Isolationsmodus aus, um festzulegen, welche Dateien und Registrierungsschlüssel sichtbar sind und von der virtuellen Anwendung, die Sie erstellen, geschrieben werden.

Option	Aktion
Merged (Zusammengeführt)	Erlaubt der Anwendung, Ressourcen auf dem lokalen Computer zu lesen und auf ihn zu schreiben
WriteCopy	Erlaubt der Anwendung, Ressourcen auf dem lokalen Computer zu lesen und die meisten Änderungen auf die Sandbox zu beschränken. ThinApp kopiert physische Dateisystemänderungen in die Sandbox, um sicherzustellen, dass ThinApp nur Kopien anstatt der tatsächlichen Dateien verändert.

Festlegen der Einstellungen für die Anwendungs-Auslieferung

Sie können das Dateiformat einer Anwendung, die Größe des Pakets und den Speicherort eines ThinApp-Projekts festlegen. Das Projekt besteht aus den Daten, die beim Kapselungsvorgang erstellt wurden. Sie können die gekapselte Anwendung erst ausführen oder bereitstellen, nachdem Sie aus den Projektdateien ein Paket erstellt haben. Das Paket ist die ausführbare Datei oder die MSI-Datei mit ausführbaren Dateien, mit der Sie eine gekapselte Anwendung ausführen oder bereitstellen.

Eine typische Firefox-Anwendung benötigt keine MSI-Installation. Andere Anwendungen jedoch, beispielsweise Microsoft Office, die sich in Anwendungs-Auslieferungstools integrieren, funktionieren problemlos als MSI-Paket. Für die MSI-Generierung müssen Sie MSI auf dem Zielgerät installieren, bevor Sie das Anwendungspaket verwenden können.

MSI-Pakete automatisieren den Registrierungsvorgang von Dateitypverknüpfungen, die Registrierung von Verknüpfungen auf dem Desktop und im **Start**-Menü sowie die Anzeige von Systemsteuerungserweiterungen. Wenn Sie vorhaben, mit ThinApp ausführbare Dateien direkt auf jedem Computer bereitzustellen, können Sie mit dem Dienstprogramm `thinreg.exe` die gleiche Registrierung durchführen.

Weitere Informationen über MSI-Dateien erhalten Sie unter „Erstellen einer MSI-Datenbank“ auf Seite 31.

So legen Sie die Einstellungen für die Anwendungsauslieferung im Setup Capture-Assistenten fest

1 (Optional) Ändern Sie das Verzeichnis, in dem Sie das ThinApp-Projekt speichern möchten.

   Wenn Sie das Standardverzeichnis behalten und Firefox 2.0.0.3 kapseln, wird als Pfad möglicherweise `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox (2.0.0.3)` angezeigt. Der im Assistenten gewählte Bestandsname bestimmt den Standardnamen des Projektverzeichnisses.

2 (Optional) Aktivieren Sie das Kontrollkästchen **MSI-Paket erstellen (Build MSI package)** und ändern Sie den MSI-Dateinamen.

3 (Optional) Um eine kleinere ausführbare Datei für Speicherorte wie zum Beispiel auf einem USB-Stick zu erstellen, aktivieren Sie das Kontrollkästchen **Virtuelles Paket komprimieren (Compress virtual package)**.

   Unter normalen Umständen verringert die Komprimierung den auf der Festplatte benötigten Speicherplatz um 50 Prozent, verlangsamt die Anwendungsausführung jedoch, wenn ThinApp die ersten Blöcke dekomprimiert, die die Anwendung starten.

4 Klicken Sie auf **Speichern (Save)**, um das Projekt zu erstellen.

Erstellen von virtuellen Anwendungen

Das Anwendungspaket ist die ausführbare Datei oder MSI-Datei, mit der Sie eine gekapselte Anwendung ausführen oder bereitstellen. Bevor Sie das Paket aus dem ThinApp-Projekt erstellen, können Sie die Projektdateien überprüfen und Einstellungen aktualisieren. Wenn Sie zum Beispiel Firefox 2.0.0.3 kapseln, könnte der Speicherort der Projektdateien `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3` lauten. Sie können das Projekt durchsuchen, bevor Sie die ausführbare Datei oder MSI-Datei für die Anwendung erstellen, um eine Einstellung (z. B. eine Active Directory-Einstellung) in der `Package.ini`-Datei zu aktualisieren, die die während des Kapselungsvorgangs festgelegten Parameter enthält. Information über die erweiterte Konfiguration der `Package.ini`-Datei finden Sie unter „Ändern der Einstellungen in der Package.ini-Datei“ auf Seite 23.

Das Projekt umfasst Ordner, wie `%AppData%`, die Dateisystempfade darstellen und Speicherorte verändern können, wenn sie auf verschiedenen Betriebssystemen oder Computern ausgeführt werden. Die meisten Ordner enthalten `##Attributes.ini`-Dateien, die den Isolationsmodus auf der Ordnerebene bestimmen. Die Isolationsmoduseinstellung auf der spezifischen Ordnerebene überschreibt die allgemeine Isolationsmoduseinstellung der `Package.ini`-Datei.

Wenn Sie eine Anwendung auf einem 32-Bit-Betriebssystem kapseln und auf einem 64-Bit-Betriebssystem erstellen möchten, müssen Sie die `THINSTALL_BIN`-Umgebungsvariable des Computers mit dem 64-Bit-Betriebssystem auf `C:\Programme (x86)\VMware\VMware ThinApp` setzen. Sie müssen das Paket nicht auf demselben Computer erstellen, auf dem Sie die Anwendung gekapselt haben. Sie können das Projekt auf einen anderen Computer kopieren und den Computer, mit dem die Kapselung ausgeführt wurde, ignorieren.

So erstellen Sie virtuelle Anwendungen im Setup Capture-Assistenten

1 (Optional) Klicken Sie auf der Seite **Anwendungspaket erstellen (Build Application Package)** auf **Package.ini bearbeiten (Edit Package.ini)**, um die Anwendungsparameter zu ändern.

2 (Optional) Zur Ansicht der ThinApp-Projektdateien in Windows Explorer klicken Sie auf **Projektordner öffnen (Open project folder)**.

3 (Optional) Um das Erstellen zu verhindern, aktivieren Sie das Kontrollkästchen **Erstellungsvorgang überspringen (Skip the build process)**.

   Sie können das Paket zu einem späteren Zeitpunkt mithilfe der Datei `build.bat` im gekapselten Anwendungsordner erstellen. Ein Firefox 2.0.0.3-Pfad zur `build.bat`-Datei könnte beispielsweise `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\build.bat` sein.

4 Klicken Sie auf **Erstellen (Build)**, um eine ausführbare Datei oder MSI-Datei mit den während des Setup Capture-Prozesses installierten Dateien zu erstellen.

5 Klicken Sie auf **Fertigstellen (Finish)**.

   Nachdem Sie auf **Fertigstellen (Finish)** geklickt haben, können Sie das Paket jederzeit erneut erstellen, wenn Sie Änderungen vornehmen müssen.

Erweiterte Paketkonfiguration

Fortgeschrittene Benutzer können die Konfigurationsdateien außerhalb des Setup Capture-Assistenten ändern, wie beispielsweise in den Dateien `Package.ini` oder `##Attributes.ini`.

Ändern der Einstellungen in der Package.ini-Datei

Die `Package.ini`-Datei enthält Konfigurationseinstellungen und befindet sich im Ordner der gekapselten Anwendung. Beispielsweise lautet ein möglicher Firefox 2.0.0.3-Pfad `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\Package.ini`.

Die folgenden Parameter sind Beispiele für Einstellungen, die Sie ändern können:

- `DirectoryIsolationMode` – Setzt den Isolationsmodus auf `Merged (Zusammengeführt)`, `WriteCopy` oder `Full (Voll)`.

  ThinApp speichert die Isolationsmodi für die Registrierung und das Dateisystem während der Laufzeit in der Sandbox. Wenn Sie den Isolationsmodus für das Projekt ändern und die ausführbare Datei erneut erstellen, müssen Sie möglicherweise die Sandbox löschen, bevor die Änderung wirksam wird.

- `PermittedGroups` – Beschränkt die Verwendung eines Anwendungspakets auf eine spezifische Menge von Active Directory-Benutzern.

- `SandboxName` – Benennt die ThinApp-Sandbox.

  Sie können den Namen für inkrementelle Anwendungs-Updates beibehalten und für umfassende Updates ändern.

- `SandboxPfad` – Bestimmt den Speicherort der Sandbox.

  Sie können die Sandbox auf einen USB-Speicherort festlegen, wenn sich die ausführbare Datei dort befindet.

- `SandboxNetworkDrives` – Gibt an, ob direkte Schreibvorgänge von der Netzwerkfreigabe in die Sandbox geleitet werden.

- `RequiredAppLinks` – Gibt eine Liste mit externen ThinApp-Paketen an, die zur Laufzeit in das aktuelle Paket importiert werden sollen.

  Kann ThinApp ein Paket nicht importieren, so beendet das Programm die Basisanwendung.

- `OptionalAppLinks` – Gibt eine Liste mit externen ThinApp-Paketen an, die zur Laufzeit in das aktuelle Paket importiert werden sollen.

  Kann ThinApp ein Paket nicht importieren, so lässt das Programm den Start der Basisanwendung zu.

Allgemeine Informationen über alle `Package.ini`-Parameter erhalten Sie unter Anhang A, „Konfigurieren von Paketparametern“, auf Seite 65.

Bearbeiten der Package.ini-Datei

Verwenden Sie einen Text-Editor, um die `Package.ini`-Datei zu aktualisieren.

So bearbeiten Sie die Package.ini-Parameter

1 Öffnen Sie die `Package.ini`-Datei, die sich im Ordner der gekapselten Anwendung befindet.

Beispielsweise lautet ein möglicher Firefox 2.0.0.3-Pfad `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\Package.ini`.

2 Aktivieren Sie den Parameter, den Sie bearbeiten möchten, indem Sie das Semikolon am Anfang der Zeile entfernen.

Aktivieren Sie beispielsweise für Firefox den `RemoveSandboxOnExit`-Parameter.

```
RemoveSandboxOnExit=1
```

Ein weiteres Beispiel könnte das Auskommentieren des `Protocols`-Parameters sein, wenn Sie nicht möchten, dass Firefox die Protokolle übernimmt.

3 Löschen oder ändern Sie den Wert des Parameters und speichern Sie die Datei.

4 Doppelklicken Sie auf die `build.bat`-Datei im Ordner der gekapselten Anwendung, um das Anwendungspaket neu zu erstellen.

Ein Firefox 2.0.0.3-Pfad zur `build.bat`-Datei könnte beispielsweise `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\build.bat` sein.

Ändern der Einstellungen in der ##Attributes.ini-Datei

Die `##Attributes.ini`-Datei wendet Konfigurationseinstellungen auf der Verzeichnisebene an. Die `Package.ini`-Datei wendet Einstellungen auf der gesamten Anwendungsebene an.

Sie können zum Beispiel den Isolationsmodus auf der Verzeichnisebene oder auf der Anwendungsebene festlegen, um zu bestimmen, welche Dateien und Registrierungsschlüssel durch die von Ihnen erstellte virtuelle Anwendung gelesen oder geschrieben werden können. Die detaillierte Einstellung in der `##Attributes.ini`-Datei setzt die allgemeine `Package.ini`-Einstellung außer Kraft. Die `Package.ini`-Einstellung legt den Isolationsmodus nur dann fest, wenn ThinApp keine `##Attributes.ini`-Informationen hat.

Um nur einzelne Ordner mit großen Dateien statt einer ganzen Anwendung zu komprimieren, können Sie Dateien auf der Ordnerebene mit dem `CompressionType`-Parameter in der `##Attributes.ini`-Datei komprimieren.

Die `##Attributes.ini`-Datei wird in den meisten Ordnern der gekapselten Anwendung angezeigt. Zum Beispiel könnte die `Attributes.ini`-Datei unter `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\%AppData%\##Attributes.ini` gespeichert sein.

Bearbeiten der ##Attributes.ini-Datei

Aktualisieren Sie die `##Attributes.ini`-Datei mithilfe eines Text-Editors.

So bearbeiten Sie die ##Attributes.ini-Parameter

1 In der `##Attributes.ini`-Datei können Sie Parameter auskommentieren, aktualisieren oder löschen.

2 Doppelklicken Sie auf die `build.bat`-Datei im Ordner der gekapselten Anwendung, um das Anwendungspaket neu zu erstellen.

Ändern der Isolationsmodi

ThinApp stellt die Auswahl der Isolationsmodi „Merged“ und „WriteCopy“ im Setup Capture-Assistenten zur Verfügung. Informationen über diese Modi erhalten Sie unter „Festlegen des Dateisystemzugriffs“ auf Seite 19.

Sie können einen dritten Isolationsmodus, „Full“, außerhalb des Assistenten in den ThinApp-Konfigurationsdateien verwenden. Der Isolationsmodus „Full (Voll)“ sichert die virtuelle Umgebung, indem er die Sichtbarkeit für Systemelemente außerhalb des virtuellen Anwendungspakets blockiert. Dieser Modus beschränkt Änderungen an Dateien oder Registrierungsschlüsseln auf die Sandbox und stellt sicher, dass keine Interaktion mit der Umgebung außerhalb des virtuellen Anwendungspakets stattfindet. Der Isolationsmodus „Full (Voll)“ verhindert Anwendungskonflikte zwischen der virtuellen Anwendung und den auf dem physischen System installierten Anwendungen.

ThinApp speichert die Isolationsmodi für die Registrierung und das Dateisystem während der Laufzeit in der Sandbox. Wenn Sie den Isolationsmodus für das Projekt ändern und die ausführbare Datei erneut erstellen, müssen Sie möglicherweise die Sandbox löschen, bevor die Änderung wirksam wird.

Sie können Isolationsmodi in der `Package.ini`-Datei und der `##Attributes.ini`-Datei ändern. Siehe „Bearbeiten der Package.ini-Datei“ auf Seite 24 und „Bearbeiten der ##Attributes.ini-Datei“ auf Seite 24. Informationen über die Auswirkung von Anwendungs-Updates auf die Isolationsmodi erhalten Sie unter „Beeinflussen der Isolationsmodi mit Application Link“ auf Seite 46.

Bereitstellen von Anwendungen 3

Für das Arbeiten mit gekapselten Anwendungen könnte die Verwendung von Bereitstellungstools, des Dienstprogramms `thinreg.exe`, MSI-Dateien sowie des Verzeichnisdienstes Active Directory erforderlich sein.

Dieser Abschnitt umfasst die folgenden Themen:

- „ThinApp-Bereitstellungsoptionen“ auf Seite 27
- „Einrichten der Dateitypzuordnungen mit dem Dienstprogramm thinreg.exe“ auf Seite 28
- „Erstellen einer MSI-Datenbank“ auf Seite 31
- „Steuern des Anwendungszugriffs mit Active Directory“ auf Seite 33
- „Verwenden von ThinApp-Paketen mit Netzwerkstreaming“ auf Seite 34
- „Verwenden von gekapselten Anwendungen mit anderen Systemkomponenten“ auf Seite 37
- „Beispielkonfiguration für Isolationsmodus je nach Bereitstellungskontext“ auf Seite 39

ThinApp-Bereitstellungsoptionen

Sie können gekapselte Anwendungen mit Bereitstellungstools, in einer VMware View-Umgebung, auf einer Netzwerkfreigabe oder als grundlegende ausführbare Dateien bereitstellen.

Bereitstellung von ThinApp mit Bereitstellungstools

Mittelgroße und große Unternehmen verwenden häufig größere Bereitstellungstools, z. B. von Symantec, BMC und SMS. ThinApp funktioniert mit allen größeren Bereitstellungstools.

Wenn Sie eines dieser Tools verwenden, können Sie MSI-Dateien für die gekapselten Anwendungen erstellen und genau so vorgehen wie bei der Bereitstellung von nativen MSI-Dateien. Siehe die Bereitstellungsanleitungen der entsprechenden Händler. Informationen über MSI-Dateien erhalten Sie unter „Erstellen einer MSI-Datenbank“ auf Seite 31.

Bereitstellen von ThinApp in der VMware View-Umgebung

Wenn Sie mit VMware View arbeiten, besteht der Arbeitsablauf aus folgenden Aufgaben:

- Erstellen von ausführbaren Dateien für die gekapselten Anwendungen.
- Speichern der ausführbaren Dateien auf einer Netzwerkfreigabe.
- Erstellen eines Anmeldeskripts, das Anwendungen, für die der Anwender berechtigt ist, abfragt und das Dienstprogramm `thinreg.exe` mit der Option ausführt, die die Anwendung auf dem lokalen Computer registriert. Anmeldeskripts sind insbesondere für nicht persistente Desktops nützlich. Siehe „Einrichten der Dateitypzuordnungen mit dem Dienstprogramm thinreg.exe“ auf Seite 28.

- Steuerung des Benutzerzugriffs auf Dateifreigaben. IT-Administratoren möchten möglicherweise den Zugriff steuern, indem sie Netzwerkfreigaben nach Funktionen organisieren und die Berechtigung zum Zugriff auf Netzwerkfreigaben anhand dieser Funktionsbegrenzungen zuweisen.

Bereitstellen von ThinApp auf Netzwerkfreigaben

Kleine und mittelständische Unternehmen setzen häufig Netzwerkfreigaben ein. Sie können ausführbare Dateien für die gekapselte Anwendung erstellen und diese Dateien auf einer Netzwerkfreigabe speichern. Nun können Sie jedes Mal, wenn Sie eine neue Anwendung oder ein Update für ein vorhandenes Paket bereitstellen, Client-Benutzer benachrichtigen, damit sie das Dienstprogramm `thinreg.exe` mit einer entsprechenden Option ausführen.

IT-Administratoren können den Zugriff von Benutzern auf Dateifreigaben steuern, indem sie Netzwerkfreigaben nach Funktionen organisieren und die Berechtigung zum Zugriff auf Netzwerkfreigaben anhand dieser Funktionsbegrenzungen zuweisen.

Die Unterschiede zwischen den Optionen Netzwerkfreigabe und VMware View liegen darin, dass die Netzwerkfreigabe von einer Mischung aus physischen und virtuellen (persistenten) Desktops ausgeht und von den Benutzern erwartet wird, dass sie das Dienstprogramm `thinreg.exe` ausführen, anstatt sich auf Anmeldeskripts zu verlassen.

Bereitstellen von ThinApp unter Verwendung von ausführbaren Dateien

Diese grundlegende Option können Sie in einer Umgebung verwenden, in der die Festplattenbelegung eingeschränkt ist. Sie können ausführbare Dateien für die gekapselten Anwendungen erstellen, sie von einem zentralen Repository kopieren und das Dienstprogramm `thinreg.exe` manuell ausführen, um die Dateitypenzuordnungen, Desktop-Verknüpfungen und das Anwendungspaket auf dem System zu registrieren.

Einrichten der Dateitypzuordnungen mit dem Dienstprogramm thinreg.exe

ThinApp erfordert, dass Sie das Dienstprogramm `thinreg.exe` zum Öffnen von Dateien, wie beispielsweise `.doc`-Dokumenten oder `.html`-Seiten verwenden. Wenn Sie z. B. auf eine URL in einer E-Mail-Nachricht klicken, muss ThinApp so eingerichtet sein, dass Firefox geöffnet wird. Für MSI-Dateien müssen Sie das Dienstprogramm `thinreg.exe` nicht ausführen, weil MSI-Dateien das Dienstprogramm bereits während der Installation der Anwendung automatisch starten.

Das Dienstprogramm `thinreg.exe` erstellt das **Start**-Menü und die Desktop-Verknüpfungen, richtet Dateitypzuordnungen ein, fügt der Systemsteuerung Informationen zum Deinstallieren des Programms hinzu und entfernt die Registrierung von zuvor registrierten Paketen. Mit dem Dienstprogramm können Sie außerdem die Systemsteuerungserweiterungen für Anwendungen anzeigen, wie beispielsweise QuickTime, oder das Systemsteuerungs-Applet Mail für Microsoft Outlook 2007. Wenn Sie mit der rechten Maustaste auf eine `.doc`-Datei klicken, können Sie mit dem Dienstprogramm `thinreg.exe` dieselben Menüoptionen für eine `.doc`-Datei in einer nativen Umgebung anzeigen.

Wenn eine Anwendung SMTP- oder HTTP-Protokolle ausführt, wie beispielsweise einen E-Mail-Link auf einer Website, der in Microsoft Outlook 2007 geöffnet werden muss, startet das Dienstprogramm `thinreg.exe` verfügbare virtuelle Anwendungen, die diese Protokolle verarbeiten können. Falls keine virtuellen Anwendungen zur Verfügung stehen, startet das Dienstprogramm `thinreg.exe` native Anwendungen, die diese Protokolle verarbeiten können.

Der Standard-Speicherort des Dienstprogramms lautet `C:\Programme\VMware\VMware ThinApp`.

Auswirkung von Application Sync auf das Dienstprogramm thinreg.exe

ThinApp stellt das Dienstprogramm Application Sync zum Update eines Anwendungspakets zur Verfügung. Das Dienstprogramm Application Sync wirkt sich folgendermaßen auf das Dienstprogramm `thinreg.exe` aus:

- Wenn Sie ausführbare Dateien hinzufügen, ändern oder entfernen, registriert das Dienstprogramm `thinreg.exe` die Dateitypzuordnungen, Verknüpfungen und Symbole erneut.
- Wenn Sie Protokolle, MIME-Typen, Systemsteuerungs-Applets und Vorlagen, die keine ausführbaren Dateien sind, installieren, registriert das Dienstprogramm `thinreg.exe` diese Elemente erneut.

Informationen über das Dienstprogramm Application Sync erhalten Sie unter „Application Sync-Updates“ auf Seite 41.

Ausführen des Dienstprogramms thinreg.exe

Dieses Beispiel bietet einige `thinreg.exe`-Befehle zur Veranschaulichung. Der Paketname in den `thinreg.exe`-Befehlen kann auf die folgende Art und Weise angezeigt werden:

- `C:\<Absoluter_Pfad_zu_.exe>`
- Relativer Pfad zur `.exe`-Datei
- `\\<Server>\<Freigabe>\<Pfad_zu_.exe>`

  Als Variation können Sie eine Platzhalterspezifikation verwenden, wie beispielsweise `*.exe`.

  Falls der Pfad oder Dateiname Leerstellen enthält, schließen Sie den Pfad in doppelte Anführungszeichen ein. Der folgende Befehl zeigt die Verwendung von doppelten Anführungszeichen.

  ```
  thinreg.exe "\\DEPLOYSERVER\ThinApps\Microsoft Office Word 2007.exe"
  ```

Informationen über `thinreg.exe`-Parameter erhalten Sie unter „Optionale thinreg.exe-Parameter“ auf Seite 29.

So führen Sie das Dienstprogramm thinreg.exe aus

1 Bestimmen Sie die ausführbaren Dateien, die ThinApp in der lokalen Umgebung registrieren muss.

2 Geben Sie in der Befehlszeile den Befehl `thinreg.exe` ein.

  ```
  thinreg.exe [<Optionale_Parameter>]
  [<package1.exe>][<package2.exe>][<Pakete_nach_Platzhalter>]
  ```

  Falls der Servername DEPLOYSERVER und die Freigabe `ThinApps` lautet, verwenden Sie das folgende Beispiel, um Microsoft Word für den angemeldeten Benutzer zu registrieren.

  ```
  ThinReg.exe "\\DEPLOYSERVER\ThinApps\Microsoft Office 2007 Word.exe"
  ```

  Verwenden Sie das folgende Beispiel, um alle Microsoft Office-Anwendungen in dem angegebenen Verzeichnis für den angemeldeten Benutzer zu registrieren.

  ```
  ThinReg.exe "\\DEPLOYSERVER\ThinApps\Microsoft Office *.exe"
  ```

Optionale thinreg.exe-Parameter

Das Dienstprogramm `thinreg.exe` überwacht die Einstellung `PermittedGroups` in der Datei `Package.ini` und registriert bzw. entfernt die Registrierung nach Bedarf. Wenn das Dienstprogramm `thinreg.exe` ein Paket für den aktuellen Benutzer registriert, erstellt das Dienstprogramm nur die Verknüpfungen und die Dateitypzuordnungen, für die der aktuelle Benutzer in der Einstellung `PermittedGroups` berechtigt ist. Falls diese Einstellung nicht vorhanden ist, ist der aktuelle Benutzer für alle ausführbaren Dateien berechtigt.

Wenn das Dienstprogramm `thinreg.exe` ein Paket für alle Benutzer mit dem Parameter `/allusers` registriert, erstellt ThinApp alle Verknüpfungen und Dateitypzuordnungen, unabhängig von der Einstellung in `PermittedGroups`. Wenn Sie auf ein Verknüpfungssymbol klicken, für das Sie keine Berechtigung besitzen, können Sie die Anwendung nicht ausführen.

Enthält der Paketname, den Sie registrieren bzw. für den Sie die Registrierung entfernen möchten, Leerstellen, so müssen Sie den Paketnamen in doppelte Anführungszeichen einschließen.

Informationen über die Einstellung `PermittedGroups` sowie Unterstützung für Active Directory-Gruppen erhalten Sie unter „PermittedGroups“ auf Seite 72.

Tabelle 3-1 listet optionale Parameter für das Dienstprogramm `thinreg.exe` auf. Für alle Befehle, die den Parameter `/a` verwenden, sind Administratorrechte erforderlich.

Tabelle 3-1. Optionale thinreg.exe-Parameter

Parameter	Zweck	Beispiel für die Verwendung
`/a`, `/allusers`	Registriert ein Paket für alle Benutzer. Falls ein nicht autorisierter Benutzer versucht, die Anwendung auszuführen, wird dieser durch eine Meldung informiert, dass er die Anwendung nicht ausführen kann.	`thinreg.exe /a "\\<Server>\<Freigabe>\Microsoft Office 2007 Word.exe"`
`/q`, `/quiet`	Verhindert die Anzeige einer Fehlermeldung für einen nicht erkannten Befehlszeilenparameter.	`thinreg.exe /q <Unbekannte_Option>`

`/u`, `/unregister`, `/uninstall`	Entfernt die Registrierung für ein Paket.
	Mit diesem Befehl wird die Software aus dem Systemsteuerungs-Applet „Software“ entfernt.
		Registrierung für Microsoft Word für den aktuellen Benutzer entfernen.
		`thinreg.exe /u "\\<Server>\<Freigabe>\Microsoft Office 2007 Word.exe"`
		Registrierung für alle Microsoft Office-Anwendungen für den aktuellen Benutzer entfernen und den Eintrag unter „Software“ entfernen.
		`thinreg.exe /u "\\server\share\Microsoft Office *.exe"`
		Falls ein Benutzer das Paket mit dem Parameter `/a` registriert, müssen Sie den Parameter `/a` auch beim Aufheben der Registrierung des Pakets verwenden.
		`thinreg.exe /u /a *.exe`

`/r`, `/reregister`	Registriert ein Paket erneut.
	Unter normalen Umständen kann das Dienstprogramm `thinreg.exe` erkennen, ob ein Paket bereits registriert ist, und es überspringen. Mit der Option `/r` wird das Dienstprogramm `thinreg.exe` dazu gezwungen, das Paket erneut zu registrieren.
		`thinreg.exe /r "\\<Server>\<Freigabe>\Microsoft Office 2007 Word.exe"`
		Falls ein Benutzer das Paket mit dem Parameter `/a` registriert, müssen Sie den Parameter `/a` auch bei der Neuregistrierung des Pakets verwenden.
		`thinreg.exe /r /a *.exe`

`/k`, `/keepunauthorized`, `/keep`	Verhindert das Entfernen der Registrierungsinformationen, auch wenn Sie keine Berechtigung mehr zum Zugriff auf ein Anwendungspaket besitzen. Ohne diese Option entfernt das Dienstprogramm `thinreg.exe` die Registrierungsinformationen für dieses Paket, wenn es erkennt, dass Sie keine Berechtigung mehr zum Zugriff auf das Paket besitzen. ThinApp speichert die Autorisierungsinformationen im Parameter `PermittedGroups` der `Package.ini`-Datei.	`thinreg.exe /k "\\<Server>\<Freigabe>\Microsoft Office 2007 Word.exe"`

Tabelle 3-1. Optionale thinreg.exe-Parameter (Fortsetzung)

Parameter	Zweck	Beispiel für die Verwendung
`/noarp`	Verhindert das Erstellen eines Eintrags in dem Systemsteuerungs-Applet „Software“.	`thinreg.exe /q /noarp "\\<Server>\<Freigabe>\Microsoft Office 2007 Word.exe"`
`/norelaunch`	Startet das Dienstprogramm `thinreg.exe` auf Microsoft Vista ohne erhöhte Rechte. Standardbenutzer können das Dienstprogramm ohne ein Popup-Fenster für die Benutzerkontensteuerung (User Account Control, UAC) starten. Wenn die Notwendigkeit für weitere Rechte mit dem Dienstprogramm `thinreg.exe` erkannt wird, wie beispielsweise die für den Parameter `/allusers` erforderlichen Berechtigungen, wird das Dienstprogramm erneut gestartet und generiert ein Popup-Fenster für die UAC. Die Option `/norelaunch` blockiert diesen Neustartvorgang und führt zu einer fehlgeschlagenen Registrierung.	`thinreg.exe /q /norelaunch "\\<Server>\<Freigabe>\Microsoft Office 2007 Word.exe"`

Erstellen einer MSI-Datenbank

Wenn Sie keine MSI-Dateien mit dem Setup Capture-Assistenten erstellen, können Sie diese Dateien nach der Generierung einer Anwendung erstellen. Eine MSI-Datenbank ist nützlich für die Verteilung von gekapselten Anwendungen über herkömmliche Desktop-Managementsysteme an Remotestandorte und zur automatischen Erstellung von Verknüpfungen und Dateitypzuordnungen. Grundlegende Active Directory-Gruppenrichtlinien bieten Möglichkeiten zur Verteilung und zum Starten von MSI-Paketen.

ThinApp erstellt eine MSI-Datenbank, die je nach der Größe der Datenbank bestimmte Dateien enthält:

- Für Datenbanken, die kleiner als 2 GB sind, besteht die MSI-Datenbank aus gekapselten ausführbaren Dateien, der Installationslogik und dem Dienstprogramm `thinreg.exe`.
- Für Datenbanken, die größer als 2 GB sind, besteht die MSI-Datenbank aus der Installationslogik und dem Dienstprogramm `thinreg.exe`. ThinApp speichert die gekapselten ausführbaren Dateien in Kabinettdateien. Diese Dateien lauten beispielsweise `<Betand_Name>_1.CAB` und `<Betand_Name>_2.CAB`. Die `.CAB`-Dateien müssen sich im selben Verzeichnis wie die MSI-Dateien befinden. ThinApp muss diese Dateien zusammen mit der MSI-Datei verteilen, um eine vollständige Installation zu ermöglichen.

Anpassen von MSI-Dateien mit Package.ini-Parametern

Sie können das Verhalten von MSI-Dateien anpassen, indem Sie die `Package.ini`-Parameter anpassen, wie beispielsweise die folgenden Parameter, und das Anwendungspaket neu erstellen:

- Der Parameter `MSIInstallDirectory` richtet das Installationsverzeichnis für das Paket ein.

  Schließen Sie beispielsweise `MSIInstallDirectory=C:\Programme\` in die `Package.ini`-Datei ein.

- Der Parameter `MSIDefaultInstallAllUsers` richtet die Installation des Pakets für individuelle Benutzer ein. ThinApp installiert das Paket im Benutzerverzeichnis `%AppData%`.

  Schließen Sie beispielsweise `MSIDefaultInstallAllUsers=0` in die `Package.ini`-Datei ein.

  Weitere Informationen über diesen Parameter erhalten Sie unter „Festlegen einer Datenbankinstallation für einzelne Benutzer und Rechner“ auf Seite 32.

- Der Parameter `MSIFileName` vergibt einen Namen für das Paket.

  Schließen Sie beispielsweise `MSIFilename=Firefox30.msi` in die `Package.ini`-Datei ein.

- Der Parameter `MSIRequireElevatedPrivileges` gibt an, ob das Installationsprogramm erhöhte Rechte für die Bereitstellung auf Microsoft Vista benötigt. Installationen für individuelle Benutzer erfordern normalerweise keine erhöhten Rechte, aber Installationen, die für den ganzen Rechner erfolgen, benötigen solche Rechte.

  Schließen Sie beispielsweise `MSIRequireElevatedPrivileges=1` in die `Package.ini`-Datei ein.

- Der Parameter `MSIProductCode` vereinfacht die Installation einer neuen Version der Anwendung. Eine MSI-Datenbank enthält einen Produktcode und einen Upgrade-Code. Wenn Sie ein Paket aktualisieren, behalten Sie den ursprünglichen Wert des Parameters `MSIUpgradeCode` bei.

  Ist der Parameterwert der neuen Version derselbe wie der Wert der alten Version, so werden Sie von der Installation dazu aufgefordert, die alte Version zu entfernen. Wenn die Werte für den Parameter unterschiedlich sind, wird die alte Version deinstalliert und die neue Version installiert.

  VMware empfiehlt, dass Sie keinen Wert für den `MSIProductCode` festlegen und stattdessen ThinApp erlauben, für jede Version einen eigenen Produktcode zu generieren.

Unabhängig von den Parameterwerten, die beim Erstellungszeitpunkt festgelegt wurden, können Sie diese Einstellungen zum Zeitpunkt der Bereitstellung überschreiben. Siehe „Erzwingen von MSI-Bereitstellungen für jeden Benutzer oder jeden Rechner“ auf Seite 33. Weitere Informationen über MSI-Parameter erhalten Sie unter „Konfigurieren von MSI-Dateien“ auf Seite 94.

Ändern der Datei Package.ini zum Erstellen von MSI-Dateien

Zum Generieren von MSI-Dateien müssen Sie einen Wert für den Parameter `MSIFilename` eingeben. Weitere Informationen über MSI-Parameter erhalten Sie unter „Anpassen von MSI-Dateien mit Package.ini-Parametern“ auf Seite 31 und „Konfigurieren von MSI-Dateien“ auf Seite 94.

So ändern Sie die MSI-Parameter

1 Geben Sie in der `Package.ini`-Datei den MSI-Dateinamen ein.

```
MSIFilename=<Dateiname>.msi
```

Der Dateiname für Firefox könnte beispielsweise `Mozilla Firefox 2.0.0.3.msi` lauten.

2 (Optional) Aktualisieren Sie die weiteren MSI-Parameter.

3 Doppelklicken Sie auf die `build.bat`-Datei im Ordner der gekapselten Anwendung, um das Anwendungspaket neu zu erstellen.

Festlegen einer Datenbankinstallation für einzelne Benutzer und Rechner

ThinApp installiert die MSI-Datenbank über alle Rechner hinweg. Sie können die Standardinstallation mit den folgenden Parameterwerten ändern:

- Um eine Datenbankinstallation für einzelne Benutzer zu erstellen, verwenden Sie den Wert 0 für den Parameter `MSIDefaultInstallAllUsers` in der `Package.ini`-Datei. Dieser Wert erstellt `msiexec`-Parameter für jeden Benutzer.

- Um eine Datenbankinstallation für einzelne Rechner für Administratoren und einzelne Benutzerinstallationen für andere Benutzer zu erstellen, verwenden Sie den Wert 2 für den Parameter `MSIDefaultInstallAllUsers`. Administratoren gehören zur Active Directory-Gruppe der Administratoren.

Weitere Informationen über den Parameter `MSIDefaultInstallAllUsers` erhalten Sie unter „MSIDefaultInstallAllUsers“ auf Seite 95.

Erzwingen von MSI-Bereitstellungen für jeden Benutzer oder jeden Rechner

Unabhängig von den Parameterwerten, die beim Erstellungszeitpunkt festgelegt wurden, können Sie diese Einstellungen zum Zeitpunkt der Bereitstellung überschreiben. Wenn Sie beispielsweise die Datenbank mit einem Wert 1 für den Parameter `MSIDefaultInstallAllUsers` erstellt haben, können Sie trotzdem individuelle Benutzerbereitstellungen für Firefox 3.0 erzwingen, indem Sie den Befehl `msiexec /i Firefox30.msi ALLUSERS=""` eingeben.

Wenn Sie das Argument `ALLUSERS=""` für den Befehl `msiexec` verwenden, extrahiert ThinApp die gekapselten ausführbaren Dateien in das Benutzerverzeichnis `%AppData%`.

So erzwingen Sie MSI-Bereitstellungen für einzelne Benutzer

Geben Sie in der Befehlszeile den Befehl `msiexec /i <Datenbank>.msi ALLUSERS=""` ein.

So erzwingen Sie MSI-Bereitstellungen für alle Benutzer eines Rechners

Geben Sie in der Befehlszeile den Befehl `msiexec /i <Datenbank>.msi ALLUSERS=1` ein.

Überschreiben des MSI-Installationsverzeichnisses

Wenn ThinApp eine MSI-Bereitstellung für einen einzelnen Rechner durchführt, ist das Standard-Installationsverzeichnis das lokalisierte Äquivalent von `%ProgramFilesDir%\<Betand_Name> (VMware ThinApp)`. Wenn Sie ein Firefox-Paket für jeden Rechner installieren, befindet sich das Paket auf `%ProgramFilesDir%\Mozilla Firefox (VMware ThinApp)`.

Wenn ThinApp eine MSI-Bereitstellung für einzelne Benutzer ausführt, lautet das Standard-Installationsverzeichnis `%AppData%\<Betand_Name> (VMware ThinApp)`.

In beiden Fällen können Sie das Installationsverzeichnis überschreiben, indem Sie die Eigenschaft INSTALLDIR zum Befehl `msiexec` übergeben.

So überschreiben Sie das MSI-Installationsverzeichnis

Geben Sie in der Befehlszeile den Befehl `msiexec /i <Datenbank>.msi INSTALLDIR=C:\<Mein_Verzeichnis>\<Mein_Packet>` ein.

Bereitstellen von MSI-Dateien auf Microsoft Vista

Wenn Sie MSI-Dateien auf Vista bereitstellen, müssen Sie angeben, ob ein Installationsprogramm erhöhte Rechte erhalten muss. Typische Installationen für einzelne Benutzer erfordern keine erhöhten Rechte, aber Installationen, die für den gesamten Rechner erfolgen, benötigen solche Rechte. ThinApp stellt den Parameter `MSIRequireElevatedPrivileges` in der `Package.ini`-Datei zur Verfügung, der die Anforderung erhöhter Rechte festlegt, wenn der Wert auf 1 gesetzt ist. Wenn Sie den Wert 1 für diesen Parameter festlegen oder eine einzelne Benutzerinstallation über die Befehlszeile erzwingen, können hierdurch UAC-Eingabeaufforderungen generiert werden. Wenn Sie den Wert 0 für diesen Parameter festlegen, werden die UAC-Eingabeaufforderungen verhindert, aber die Bereitstellung für rechnerweite Installationen schlägt fehl.

Steuern des Anwendungszugriffs mit Active Directory

Sie können den Zugriff auf Anwendungen mithilfe von Active Directory-Gruppen steuern. Wenn Sie ein Paket erstellen, konvertiert ThinApp die Active Directory-Gruppennamen in SID-Werte. Eine SID (Security Identifier) ist ein kleiner binärer Wert, der ein Objekt eindeutig identifiziert. SID-Werte sind für einige Gruppen, wie die Administratorengruppe, nicht eindeutig. Da ThinApp die SID-Werte für die zukünftige Validierung in Paketen speichert, muss bei der Verwendung von Active Directory Folgendes in Betracht gezogen werden:

- Sie müssen während des Erstellungsvorgangs mit Ihrer Active Directory-Domäne verbunden sein und die Gruppen, die Sie festlegen, müssen vorhanden sein. ThinApp sucht während der Erstellung nach den SID-Werten.
- Wenn Sie eine Gruppe löschen und neu erstellen, könnte sich die SID ändern. Erstellen Sie in diesem Fall das Paket neu, um eine Authentifizierung der neuen Gruppe zu ermöglichen.

- Wenn Benutzer offline sind, kann ThinApp sie mithilfe von zwischengespeicherten Anmeldeinformationen authentifizieren. Wenn die Benutzer sich an ihren Rechnern anmelden können, funktioniert die Authentifizierung weiterhin. Verwenden Sie eine Gruppenrichtlinie, um den Zeitraum festzulegen, für den die zwischengespeicherten Anmeldeinformationen gültig sind.
- Zwischengespeicherte Anmeldeinformationen werden auf Clients möglicherweise erst beim nächsten Active Directory-Aktualisierungszyklus aktualisiert. Sie können eine Gruppenrichtlinie für einen Client erzwingen, indem Sie den Befehl `gpupdate` verwenden. Dieser Befehl aktualisiert die lokale Gruppenrichtlinie, die Gruppenrichtlinie sowie die Sicherheitseinstellungen, die in Active Directory gespeichert sind. Möglicherweise müssen Sie sich abmelden, bevor die Active Directory-Anmeldeinformationen erneut zwischengespeichert werden.
- Bestimmte Gruppen wie die Gruppen „Administratoren (Administratoren)“ und „Alle (Everyone)“ haben in jeder Active Directory-Domäne und -Arbeitsgruppe dieselbe SID. Andere Gruppen, die Sie erstellen, haben eine domänenspezifische SID. Benutzer können keine eigene lokale Gruppe desselben Namens erstellen, um die Authentifizierung zu umgehen.

Package.ini-Einträge für die Active Directory-Zugriffssteuerung

ThinApp stellt die Parameter `PermittedGroups` in der Datei `Package.ini` zur Verfügung, um den Zugriff auf Active Directory zu steuern. Wenn Sie eine gekapselte Anwendung starten, überprüft der Parameter `PermittedGroups`, ob der Benutzer ein Mitglied einer spezifizierten Active Directory-Gruppe ist. Falls der Benutzer noch der ein Mitglied der Active Directory-Gruppe ist, startet ThinApp die Anwendung nicht. Informationen über die Beschränkung von Paketen auf Active Directory-Gruppen erhalten Sie unter „PermittedGroups“ auf Seite 72.

Im folgenden `Package.ini`-Eintrag, erben App1 und App2 die Werte der `PermittedGroups`.

```
[BuildOptions]
PermittedGroups=Administratoren;Office–Benutzer
[App1.exe]
                ...
                ..
[App2.exe]
                ...
                ...
```

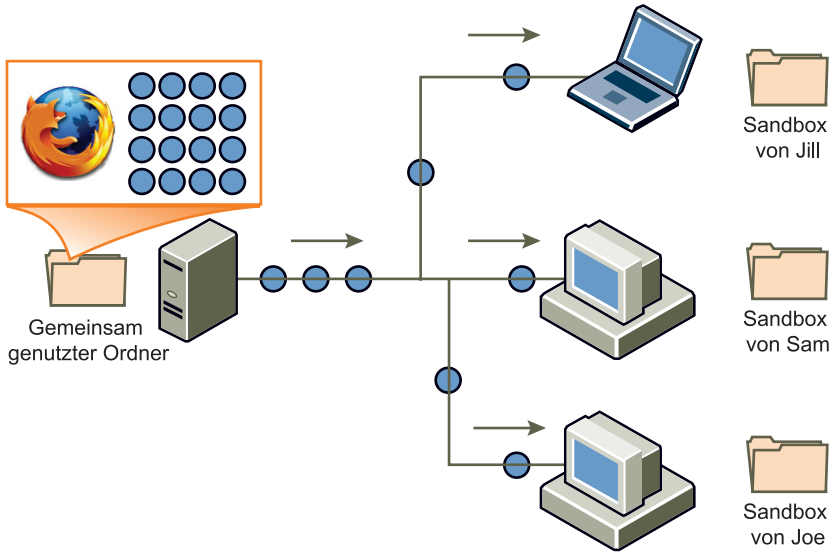
Im folgenden Eintrag können nur Benutzer, die zur Gruppe `App1–Benutzer` gehören, die Datei `App1.exe` verwenden, ist, und Mitglieder der Gruppe `Alle (Everyone)` können die Datei `App2.exe` verwenden ist. Die Standardmeldung für abgelehnte Benutzer ändert sich für App1.

```
[BuildOptions]
PermittedGroups=Alle
[App1.exe]
PermittedGroups=App1–Benutzer
AccessDeniedMsg=Sie können diese Anwendung leider nicht ausführen
..
[App2.exe]
...
...
```

Verwenden von ThinApp-Paketen mit Netzwerkstreaming

Jedes Netzwerkspeichergerät kann für Hunderte oder Tausende von Clientcomputern als Streamingserver verwendet werden. Siehe Abbildung 3-1.

Abbildung 3-1. Datenblockstreaming über eine Netzwerkfreigabe



Auf dem Desktop des Endbenutzers können Sie Verknüpfungen erstellen, die auf die zentral gehosteten ausführbaren Dateipakete verweisen. Wenn der Benutzer auf die Verknüpfung klickt, beginnt die Anwendung mit der Übertragung per Stream auf den Clientcomputer. Während des anfänglichen Streaming-Startvorgangs wird der Benutzer über die ThinApp-Statusleiste über den Fortschritt informiert.

So funktioniert Anwendungsstreaming mit ThinApp

Wenn Sie komprimierte ausführbare ThinApp-Dateien auf einer Netzwerkfreigabe oder einem USB-Flash-Laufwerk ablegen, werden die Inhalte der ausführbaren Datei im Blockformat per Stream an die Clientcomputer übertragen. Da eine Anwendung bestimmte Teile von Datendateien anfordert, liest ThinApp diese Informationen im komprimierten Format über das Netzwerk und verwendet dazu das Standardprotokoll für Dateifreigaben von Windows. Eine Ansicht des Prozesses erhalten Sie in Abbildung 3-2.

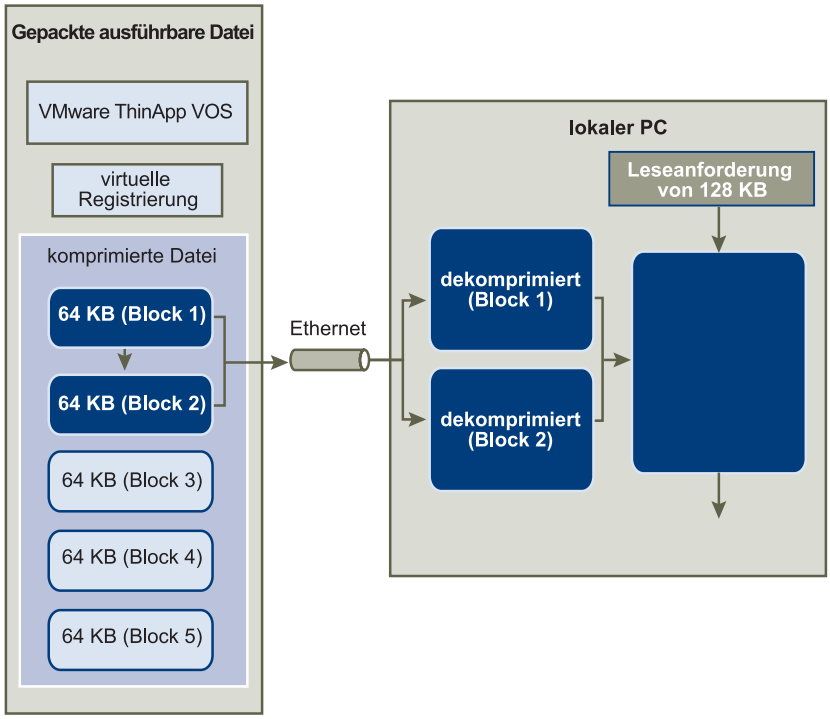
Nachdem ein Clientcomputer Daten empfangen hat, entpackt ThinApp die Daten direkt in den Arbeitsspeicher. Da ThinApp keine Daten zurück auf die Festplatte schreibt, erfolgt dieser Vorgang sehr schnell. Das Laden eines großen Pakets über das Netzwerk dauert nicht notwendigerweise lange und die Paketgröße hat keinen Einfluss auf die Startzeit einer Anwendung. Wenn Sie weitere 20 GB zu einem Paket hinzufügen, das während der Laufzeit nicht verwendet wird, wird das Paket mit der gleichen Geschwindigkeit geladen. Wenn die Anwendung geöffnet wird und 32 KB Daten von der 20 GB-Datei gelesen werden, fordert ThinApp nur 32 KB Daten an.

Der ThinApp-Laufzeitclient ist ein kleiner Bestandteil des ausführbaren Dateipakets. Wenn ThinApp den Laufzeitclient ausführt, richtet es die Umgebung ein und startet die ausführbare Zieldatei. Die ausführbare Zieldatei greift auf andere Teile der Anwendung zu, die in dem virtuellen Betriebssystem gespeichert sind. Der Laufzeitclient fängt solche Anforderungen ab und führt sie durch Laden von DLLs vom virtuellen Betriebssystem aus.

Die Ladezeit des Laufzeitclients über ein Netzwerk hinweg beträgt einige Millisekunden. Nachdem ThinApp den Laufzeitclient in den Arbeitsspeicher des Clientcomputers geladen hat, berechnet der Computer des Endbenutzers, welche Datenblöcke vom Server benötigt werden, und liest diese basierend auf der Aktivität der Anwendung.

Wenn die Anwendung anschließend Leseanforderungen für dieselben Daten durchführt, liefert der Windows-Festplatten-Cache Daten, ohne dass ein Lesevorgang auf dem Netzwerk erforderlich ist. Falls der Clientcomputer wenig Arbeitsspeicher hat, löscht Windows einen Teil des Festplatten-Caches, damit Arbeitsspeicherressourcen für andere Anwendungen zur Verfügung stehen.

Abbildung 3-2. Streaming von Anwendungen



Anforderungen und Empfehlungen für Streamingpakete

ThinApp benötigt keine spezielle Serversoftware, um die Streamingfunktion bereitstellen zu können. Jede Windows-Dateifreigabe, jedes NAS-Gerät und jede SMB-Freigabe kann diese Funktion zur Verfügung stellen. Die Datenmenge, die übertragen werden muss, bevor die Anwendung ausgeführt werden kann, ist für jede Anwendung unterschiedlich. Für Microsoft Office muss nur ein Bruchteil des Paketinhalts gestreamt werden, bevor eine Anwendung ausgeführt werden kann.

VMware empfiehlt, dass Sie ThinApp-Streaming in einer LAN-basierten Umgebung mit mindestens 100 MB-Netzwerken verwenden. Für WAN- und Internet-Bereitstellungen, bei denen häufige oder unerwartete Unterbrechungen stattfinden können, empfiehlt VMware eine der folgenden Lösungen:

- Verwenden Sie eine URL zur Bereitstellung der Anwendungen.
- Verwenden Sie eine Desktop-Bereitstellungslösung, um das Paket per Push in den Hintergrund zu bringen. Lassen Sie die Ausführung der Anwendung erst zu, nachdem das gesamte Paket heruntergeladen wurde.

Durch diese Lösungen werden Fehler verringert und Situationen beseitigt, in denen die Anwendung während eines Netzwerkausfalls nicht gestreamte Teile erfordert. Ein Unternehmen mit vielen Zweigstellen bestimmt normalerweise ein Anwendungs-Repository, das einen zentralen Freigabeordner in jeder Zweigstelle spiegelt. Mit dieser Konfiguration wird die lokale Leistung für Clientcomputer in den einzelnen Zweigstellen optimiert.

Sicherheitsempfehlungen für Streamingpakete

VMware empfiehlt, ein zentrales Freigabeverzeichnis für das Paket anzulegen und den Ordner als schreibgeschützt festzulegen. Benutzer können den jeweiligen Paketinhalt lesen, jedoch die ausführbaren Dateiinhalte nicht ändern. Wenn ein Paket von einem freigegeben Speicherort gestreamt wird, speichert ThinApp die Anwendungsänderungen in der Benutzer-Sandbox. Der Standardspeicherort für die Sandbox lautet `%AppData%\Thinstall\<Anwendungs_Name>`. Sie können den Sandbox-Speicherort während der Laufzeit oder während des Paketierens konfigurieren.

Üblicherweise erfolgt die Konfiguration derart, dass die Benutzer-Sandbox auf ein anderes zentrales Speichergerät gelegt wird. Der Benutzer kann einen beliebigen Computer verwenden und auf der zentralen Freigabe die individuellen Anwendungseinstellungen beibehalten. Wenn Pakete von einer zentralen Freigabe gestreamt werden, bleiben sie gesperrt, bis alle Benutzer die Anwendung beendet haben.

Streamen von ThinApp-Paketen über das Netzwerk

Benutzer können über das Netzwerk auf paketierte Anwendungen zugreifen.

So streamen Sie Pakete vom Netzwerk

1 Legen Sie das ThinApp-Paket an einem Speicherort ab, auf den Clientcomputer zugreifen können.

2 Senden Sie einen Link an die Benutzer, damit sie die Anwendung direkt ausführen können.

Verwenden von gekapselten Anwendungen mit anderen Systemkomponenten

Gekapselte Anwendungen können mit anderen Komponenten interagieren, die auf dem Desktop installiert sind.

Ausführen von Einfügevorgängen

Nachfolgend erhalten Sie einen Überblick über die mit ThinApp durchführbaren Einfügevorgänge sowie die für ThinApp geltenden Einschränkungen:

- **Inhalt von systemseitig installierten Anwendungen in gekapselte Anwendungen einfügen** – Dieser Einfügevorgang ist unbegrenzt. Die virtuelle Anwendung kann alle Standard-Zwischenablageformate empfangen, wie beispielsweise Text, Grafiken und HTML. Die virtuelle Anwendung kann OLE-Objekte empfangen.
- **Einfügen von gekapselten Anwendungen in Systemanwendungen** – ThinApp konvertiert OLE-Objekte, die in virtuellen Anwendungen erstellt wurden, in native Objekte, wenn Sie sie in native Anwendungen einfügen.

Zugriff auf Drucker

Eine gekapselte Anwendung hat Zugriff auf alle Drucker, die auf dem Computer installiert sind, auf dem sie ausgeführt wird. Gekapselte Anwendungen und Anwendungen, die auf dem physischen System installiert sind, haben die gleichen Druckfähigkeiten.

Sie können ThinApp nicht zur Virtualisierung von Druckertreibern verwenden. Sie müssen die Druckertreiber manuell auf einem Computer installieren.

Zugriff auf Treiber

Eine gekapselte Anwendung hat vollständigen Zugriff auf alle Gerätetreiber, die auf dem Computer installiert sind, auf dem sie ausgeführt wird. Gekapselte Anwendungen und Anwendungen, die auf dem physischen System installiert sind, haben die gleichen Beziehungen zu Gerätetreibern. Wenn eine Anwendung einen Gerätetreiber benötigt, müssen Sie diesen Treiber separat von dem ThinApp-Paket installieren.

In manchen Fällen könnte eine Anwendung ohne einen zugehörigen Treiber funktionieren, jedoch mit bestimmten Einschränkungen. Beispielsweise installiert Adobe Acrobat einen Druckertreiber, mit dem Anwendungen unter Verwendung eines Druckmechanismus systemweit PDF-Dateien erstellen können. Wenn Sie eine gekapselte Version von Adobe Acrobat verwenden, können Sie diese zum Laden, Bearbeiten und Speichern von PDF-Dateien verwenden, ohne dass die Druckertreiber installiert werden müssen. Andere Anwendungen erkennen einen neuen Druckertreiber erst dann, wenn der Treiber installiert ist.

Zugriff auf die lokale Festplatte, den Wechseldatenträger und die Netzwerkfreigaben

Wenn Sie eine Projektstruktur erstellen, konfiguriert ThinApp die Isolationsmodi für Verzeichnisse und Unterstrukturen der Registrierung. Die Isolationsmodi steuern, für welche Verzeichnisse die Anwendung auf dem lokalen Computer Lese- und Schreibberechtigungen erhält. Überblick über die Standard-Konfigurationsoptionen:

- **Festplatte** – Ein Beispiel für eine Festplatte lautet `C:\`. Isolationsmodi, die während des Kapselungsvorgangs ausgewählt werden, haben Auswirkungen auf den Zugriff. Benutzer haben Schreibberechtigungen auf ihrem `Desktop` und im Ordner `Eigene Dateien`. Andere von der Anwendung ausgeführte Änderungen werden zur Benutzer-Sandbox weitergeleitet. Der Standard-Speicherort für die Sandbox ist das Verzeichnis `Anwendungsdaten`.
- **Wechseldatenträger** – Standardmäßig hat jeder Benutzer mit Zugriffsrechten Schreib- oder Leseberechtigungen für alle Speicherorte auf einem Wechseldatenträger.
- **Netzlaufwerke** – Standardmäßig hat jeder Benutzer mit Zugriffsrechten Schreib- oder Leseberechtigungen für alle Speicherorte auf einem vom Netzwerk zugeordneten Laufwerk.
- **UNC-Netzwerkpfade** – Standardmäßig hat jeder Benutzer mit Zugriffsrechten Schreib- oder Leseberechtigungen für alle Speicherorte auf einem UNC-Netzwerkpfad.

Zugriff auf die Systemregistrierung

Standardmäßig haben gekapselte Anwendungen Leseberechtigungen für die vollständige Systemregistrierung, wie durch die Zugriffsrechte festgelegt. Bestimmte Bereiche der Registrierung werden während des Paketerstellungsvorgangs vom System isoliert. Diese Isolation verringert Konflikte zwischen verschiedenen Versionen virtueller Anwendungen und systemseitig installierter Anwendungen. Standardmäßig speichert ThinApp alle Registrierungsänderungen von gekapselten Anwendungen in einer isolierten Sandbox und das System bleibt unverändert.

Zugriff auf Netzwerk und Sockets

Gekapselte Anwendungen haben Standardzugriff auf die Netzwerkfunktionen. Gekapselte Anwendungen können eine Bindung mit lokalen Schnittstellen eingehen und Remoteverbindungen erstellen, wenn der Benutzer Zugriffsrechte zum Ausführen dieser Vorgänge besitzt.

Verwenden von gemeinsam genutztem Arbeitsspeicher und Named Pipes

Gekapselte Anwendungen können mit anderen Anwendungen im System interagieren, indem sie gemeinsam genutzten Arbeitsspeicher, Named Pipes, Mutex-Objekte und Semaphoren verwenden.

ThinApp kann gemeinsam genutzte Arbeitsspeicherobjekte und Synchronisierungsobjekte isolieren. Durch die Isolierung werden sie für andere Anwendungen unsichtbar und andere Anwendungsobjekte werden für eine gekapselte Anwendung unsichtbar.

Verwenden von COM-, DCOM- und Out-of-Process-COM-Komponenten

Gekapselte Anwendungen können COM-Steuerungen von der virtuellen Umgebung und vom System erstellen. Wenn eine COM-Steuerung als ein Out-of-Process-COM installiert wird, wird die Steuerung als virtueller Prozess ausgeführt, wenn sie von einer gekapselten Anwendung verwendet wird. Sie können die von den gekapselten Anwendungen vorgenommenen Änderungen steuern.

Starten von Diensten

Gekapselte Anwendungen können systemseitig installierte Dienste und virtuelle Dienste starten und ausführen. Systemdienste werden in der virtuellen Umgebung ausgeführt, die die von den Diensten vorgenommenen Änderungen steuert.

Verwenden von Dateitypzuordnungen

Gekapselte Anwendungen können systemseitig installierte Anwendungen ausführen, indem sie Dateitypzuordnungen verwenden. Sie können Dateitypzuordnungen zur Registrierung des lokalen Computers hinzufügen, die auf gekapselte ausführbare Dateien für einzelne Benutzer und Rechner verweisen.

Beispielkonfiguration für Isolationsmodus je nach Bereitstellungskontext

Isolationsmodi steuern den Lese- und Schreibzugriff für bestimmte Systemverzeichnisse und untergeordnete Schlüssel der Systemregistrierung. Siehe „Ändern der Isolationsmodi“ auf Seite 24.

Sie können die Isolationsmodi anpassen, um die Probleme in Tabelle 3-2 zu lösen.

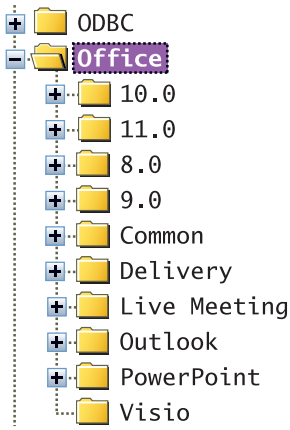
Tabelle 3-2. Beispiele für Probleme und Lösungen mithilfe von Isolationsmodi

Problem	Lösung
Eine Anwendung kann nicht ausgeführt werden, weil gleichzeitig ältere oder neuere Versionen vorhanden sind bzw. nicht ordnungsgemäß deinstalliert wurden.	Verwenden Sie den Isolationsmodus „Full (Voll)“. ThinApp verbirgt Hostcomputerdateien und Registrierungsschlüssel vor der Anwendung, wenn sich die Hostcomputerdateien in denselben Verzeichnissen und untergeordneten Schlüsseln befinden, die das Installationsprogramm der Anwendung erstellt. Für Verzeichnisse und untergeordnete Schlüssel mit dem Isolationsmodus „Full (Voll)“ erkennt die Anwendung nur virtuelle Dateien und untergeordnete Schlüssel. Alle Systemwerte, die am selben Speicherort vorhanden sind, sind für die Anwendung unsichtbar.
Eine Anwendung funktioniert nicht, weil die Benutzer sie nicht für eine Umgebung mit mehreren Benutzern konzipiert oder getestet haben. Die Anwendung kann keine Dateien und Schlüssel ändern, ohne dass dies Auswirkungen auf andere Benutzer hat.	Verwenden Sie den Isolationsmodus „WriteCopy“. ThinApp erstellt Kopien von Registrierungsschlüsseln und Dateien, die die Anwendung schreibt, und führt alle Änderungen in einer benutzerspezifischen Sandbox durch. Für Verzeichnisse und untergeordnete Schlüssel mit WriteCopy-Isolation erkennt die Anwendung die Hostcomputerdateien und virtuellen Dateien. Alle Schreibvorgänge konvertieren Hostcomputerdateien in virtuelle Dateien in der Sandbox.
Eine Anwendung funktioniert nicht, weil sie eine Schreibberechtigung für globale Speicherorte besitzt und nicht für eine gesperrte Desktop-Umgebung entworfen wurde, wie sie in Unternehmensumgebungen oder für Windows Vista üblich ist.	Verwenden Sie den Isolationsmodus „WriteCopy“. ThinApp erstellt Kopien von Registrierungsschlüsseln und Dateien, die die Anwendung schreibt, und führt alle Änderungen in einer benutzerspezifischen Sandbox durch. Für Verzeichnisse und untergeordnete Schlüssel mit WriteCopy-Isolation erkennt die Anwendung die Hostcomputerdateien und virtuellen Dateien. Alle Schreibvorgänge konvertieren Hostcomputerdateien in virtuelle Dateien in der Sandbox.

Ansicht der Auswirkung von Isolationsmodus auf die Windows-Registrierung

Abbildung 3-3 zeigt einen Bereich der Windows-Registrierung für einen Computer, auf dem ältere Microsoft Office-Anwendungen installiert sind. Microsoft Office 2003 erstellt die folgende Unterstruktur der Registrierung: `HKEY_LOCAL_MACHINE\Software\Microsoft\Office\11.0`.

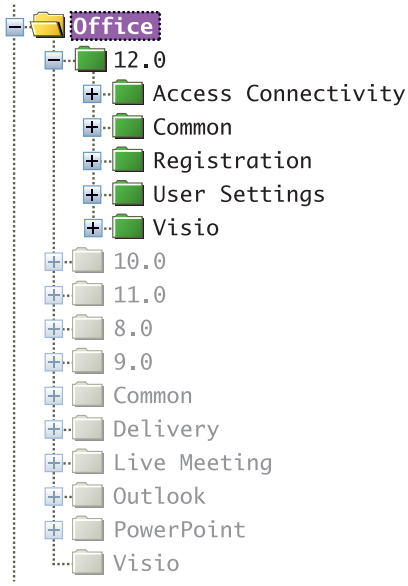
Abbildung 3-3. Die Windows-Registrierung, wie sie von Windows Regedit gesehen wird



Wenn ThinApp eine gekapselte Version von Microsoft Visio 2007 ausführt, setzt ThinApp die Unterstruktur der Registrierung `HKLM\Software\Microsoft\Office` auf den Isolationsmodus „Full (Voll)“. Mit dieser Einstellung wird verhindert, dass Microsoft Visio 2007 wegen der Registrierungseinstellungen, die möglicherweise am selben Speicherort auf dem Hostcomputer vorhanden sind, nicht funktioniert.

Abbildung 3-4 zeigt die Registrierung aus der Perspektive der gekapselten Microsoft Visio 2007-Anwendung.

Abbildung 3-4. Die Windows-Registrierung, wie sie von der gekapselten Microsoft Visio 2007-Anwendung gesehen wird



Aktualisieren von Anwendungen 4

Gekapselte Anwendungen können – abhängig vom Umfang der Änderung und den Abhängigkeiten zu anderen Anwendungen – mit unterschiedlichen Dienstprogrammen aktualisiert werden.

Dieser Abschnitt umfasst die folgenden Themen:

- „Anwendungs-Updates, die der Endbenutzer auslöst“ auf Seite 41
- „Anwendungs-Updates, die der Administrator auslöst“ auf Seite 48
- „Automatische Anwendungs-Updates“ auf Seite 50
- „Aktualisieren von laufenden Anwendungen auf einer Netzwerkfreigabe“ auf Seite 51
- „Sandbox-Überlegungen für aktualisierte Anwendungen“ auf Seite 52

Anwendungs-Updates, die der Endbenutzer auslöst

ThinApp bietet zur Aktualisierung von Anwendungen mit neuen Versionen oder neuen Komponenten die Dienstprogramme Application Sync und Application Link. Das Dienstprogramm Application Sync aktualisiert ein gesamtes Anwendungspaket. Das Dienstprogramm Application Link verwaltet gemeinsam genutzte Komponenten oder abhängige Anwendungen in separaten Paketen.

Application Sync-Updates

Mit dem Dienstprogramm Application Sync können bereitgestellte virtuelle Anwendungen auf dem neuesten Stand gehalten werden. Ist das Dienstprogramm beim Start einer Anwendung aktiviert, führt die Anwendung eine Webserverabfrage durch, um festzustellen, ob eine aktualisierte Version der ausführbaren Datei verfügbar ist. Ist ein Update verfügbar, werden die Unterschiede zwischen dem bestehenden Paket und dem neuen Paket heruntergeladen und verwendet, um eine aktualisierte Version des Pakets zu erzeugen. Bei zukünftigen Startvorgängen wird das aktualisierte Paket verwendet.

Das Dienstprogramm Application Sync ist für größere Konfigurations-Updates der Anwendung geeignet. Beispielsweise, wenn ein Update auf die nächste Hauptversion von Firefox erforderlich ist.

Verwendung von Application Sync in einer verwalteten oder nicht verwaltete Umgebung

Für virtuelle Anwendungen, die in einer verwalteten Computerumgebung automatisch aktualisiert werden, sollte Application Sync nicht verwendet werden, da das Dienstprogramm möglicherweise mit anderen Update-Funktionen Konflikte hervorruft.

Wird eine Anwendung mithilfe einer automatischen Update-Funktion aktualisiert, ist das Update in der Sandbox gespeichert. Versucht das Dienstprogramm Application Sync, die Anwendung nach einem erfolgten automatischen Anwendungs-Update zu aktualisieren, hat das in der Sandbox gespeicherte Versions-Update Vorrang vor den in der Application Sync-Version enthaltenen Dateien. Vorrang in der Reihenfolge der Dateiaktualisierungen haben die Dateien in der Sandbox, gefolgt von denen des virtuellen Betriebssystems vor denen des physischen Computers.

Verwenden Sie in einer nicht verwaltete Umgebung, in dem keinen automatischen Update der Anwendungen erfolgt, das Dienstprogramm Application Sync zum Update von Anwendungen.

Update von Firefox 2.0.0.3 auf Firefox 3 mit Application Sync

Dieses Beispiel zeigt den Update-Vorgang von Application Sync für Firefox.

Der Update-Vorgang erfordert die Bearbeitung der `Package.ini`-Datei. Der `AppSyncURL`-Parameter erfordert einen URL-Pfad. ThinApp unterstützt HTTP, HTTPS und Dateiprotokolle. Informationen über alle Parameter von Application Sync finden Sie unter „Konfigurieren von Anwendungs-Updates mit Application Sync“ auf Seite 92.

So aktualisieren Sie Firefox 2.0.0.3 auf Firefox 3

1 Kapseln Sie Firefox 2.0.0.3 und Firefox 3 in getrennte Pakete.

2 Stellen Sie sicher, dass der Name des primären Datencontainers der beiden Pakete übereinstimmt.

   Der im Setup Capture-Prozess festgelegte primäre Datencontainer ist diejenige Datei, die das virtuelle Dateisystem und die virtuelle Registrierung enthält. Wenn Sie über ein Firefox 2.0.0.3-Paket, bei dem der Name des primären Datencontainers `Mozilla Firefox 2.0.0.3.exe` lautet, und ein Firefox 3-Paket verfügen, bei dem der Name des primären Datencontainers `Mozilla Firefox 3.dat` lautet, müssen Sie den Namen im `Shortcut`-Parameter in einen gemeinsamen Namen ändern. Sie können beispielsweise `Firefox.exe` als Namen verwenden.

3 Bearbeiten Sie die `Package.ini`-Datei in jedem Paket.

   a Öffnen Sie die `Package.ini`-Datei, die sich im Ordner der gekapselten Anwendung befindet.

      Ein Pfad von Firefox 2.0.0.3 zur `Package.ini`-Datei könnte beispielsweise `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\Package.ini` sein.

   b Heben Sie die Auskommentierung der Application Sync-Parameter auf, die Sie bearbeiten möchten, indem Sie das Semikolon am Anfang der Zeile entfernen.

      Um das Dienstprogramm zu aktivieren, müssen Sie die Auskommentierung vom `AppSyncURL`-Parameter aufheben.

   c Ändern Sie den Wert der Parameter und speichern Sie die Datei.

      Sie können beispielsweise eine ausführbare Datei der neuesten Firefox-Version auf ein zugeordnetes Netzlaufwerk kopieren und als Wert für den `AppSyncURL`-Parameter einen Pfad zu diesem Speicherort eingeben. Ist `Z:` das zugeordnete Laufwerk und `Firefox` der Name des Verzeichnisses, in dem die ausführbare Datei gespeichert wird, lautet ein Beispielpfad: `file:///Z:/Firefox/Firefox.exe`.

      Stellen Sie sicher, dass der Pfad `AppSyncURL` in beiden `Package.ini`-Dateien der gleiche ist und auf die aktualisierte Version verweist.

4 Doppelklicken Sie im gekapselten Anwendungsordner auf die `build.bat`-Datei, um das Anwendungspaket erneut zu erstellen.

   Ein Firefox 2.0.0.3-Pfad zur `build.bat`-Datei könnte beispielsweise `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\build.bat` sein.

5 Um Firefox 2.0.0.3 auf Firefox 3 zu aktualisieren, starten Sie die ausführbare Datei, wie beispielsweise `Mozilla Firefox 2.0.0.3.exe`, im `\bin`-Verzeichnis.

   Wenn Sie die Anwendung vor der Ablaufzeit, die im Parameter `AppSyncExpirePeriod` der `Package.ini`-Datei eingerichtet wird, starten, lädt ThinApp das Update im Hintergrund herunter, während Sie weiterhin mit der Anwendung arbeiten. Wenn Sie die Anwendung das nächste Mal starten, sehen Sie die aktualisierte Version.

Wenn Sie die Anwendung nach Ablauf des Pakets starten, lädt ThinApp das Update im Vordergrund herunter und Sie können während dieses Zeitraums nicht mit der Anwendung arbeiten. Nach Fertigstellung des Downloads startet ThinApp die Anwendung mit der neuen Version.

Korrigieren eines Updates mithilfe von Application Sync

Wenn Sie über mehrere Download-Updates von Application Sync verfügen, beispielsweise mehrere Updates von Microsoft Office, und wenn ein bestimmtes Update sich negativ auswirkt oder entfernt werden muss, können Sie das Problem beheben.

So korrigieren Sie ein Update

Speichern Sie das richtige Update auf den Server, auf den ThinApp zugreifen kann.

Das Update wird angewendet, sobald die Anwendung das nächste Mal auf einem Clientcomputer gestartet wird.

Auswirkung von Application Sync auf ausführbare Dateien mit Einstiegspunkt

Das Dienstprogramm Application Sync aktualisiert ausführbare Dateien mit Einstiegspunkt. Angenommen, Sie stellen ein Microsoft Office 2007-Paket bereit, in dem Microsoft PowerPoint nicht enthalten ist. Der Einstiegspunkt `Microsoft Office PowerPoint 2007.exe` ist im ursprünglichen Paket nicht vorhanden. Wenn Sie das Microsoft Office 2007-Paket erneut erstellen, in dem Microsoft PowerPoint enthalten ist, und das Dienstprogramm Application Sync zum Update der Clientcomputer verwenden, können die Endbenutzer auf eine ausführbare Datei mit Einstiegspunkt für Microsoft PowerPoint zugreifen.

Aktualisieren von thinreg.exe-Registrierungen mit Application Sync

Wenn Sie mithilfe von `thinreg.exe` virtuelle Anwendungen auf Ihrem System registrieren und Anwendungen mit dem Dienstprogramm Application Sync aktualisieren, können Sie Registrierungen aktualisieren, indem Sie eine Kopie von `thinreg.exe`, die sich unter `C:\Programme\VMware\VMware ThinApp` befindet, zusammen mit dem aktualisierten Paket auf dem Server speichern.

Beibehalten des primären Datencontainernamens mit Application Sync

Das Dienstprogramm Application Sync erfordert, dass der Name des primären Datencontainers und die Datei, in der die virtuellen Dateien und die Registrierungsdaten gespeichert sind, in den alten und neuen Versionen einer Anwendung identisch sind. Sie dürfen beispielsweise keine alte Version mit `Microsoft Office Excel 2003.exe` als primären Datencontainernamen verwenden, wenn die neue Version `Microsoft Office 2007.dat` als primären Datencontainernamen verwendet. Um den Namen des primären Datencontainers zu bestätigen, prüfen Sie bitte die `ReadOnlyData`-Parameter in der `Package.ini`-Datei. Weitere Informationen über den primären Datencontainer finden Sie unter „Festlegen von Anwendungsverknüpfungen und Nachverfolgungsnamen“ auf Seite 17.

Application Link-Updates

Das Dienstprogramm Application Link verbindet abhängige Anwendungen während der Laufzeit. Sie können Komponenten getrennt paketieren, bereitstellen und aktualisieren, anstatt alle Komponenten im selben Paket zu kapseln. ThinApp unterstützt die gleichzeitige Kombination von bis zu 250 Paketen. Jedes Paket kann von beliebiger Größe sein.

Das Dienstprogramm Application Link ist für folgende Objekte nützlich:

- **Große gemeinsam genutzte Bibliotheken und Frameworks** – Verknüpfen der Laufzeit-Komponenten wie .NET, JRE oder ODBC-Treibern mit abhängigen Anwendungen.

  Sie können beispielsweise .NET mit einer Anwendung verknüpfen, selbst wenn der lokale Computer für die Anwendung die Installation von .NET nicht zulässt oder bereits über eine andere Version von .NET verfügt.

- **Zusätzliche Komponenten und Plug-Ins** – Getrenntes Verpacken und Bereitstellen der anwendungsspezifischen Komponenten und Plug-Ins von der Basisanwendung.

Sie können beispielsweise Adobe Flash Player oder Adobe Reader von einer Firefox-Basisanwendung trennen und die Komponenten verknüpfen.

Das Dienstprogramm Application Link ermöglicht Ihnen die Bereitstellung eines einzigen virtualisierten Microsoft Office-Pakets für alle Benutzer sowie die Bereitstellung individueller zusätzlicher Komponenten für einzelne Benutzer.

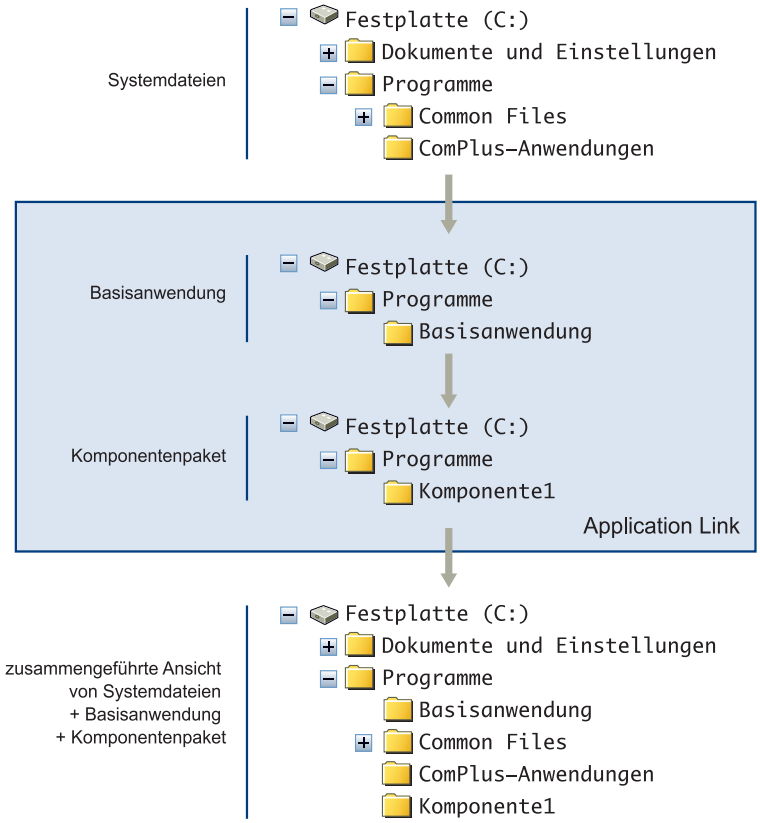
- **Hotfixes und Service Packs** – Verknüpfen der Updates mit einer Anwendung und Rollback auf eine frühere Version, wenn Benutzer mit der neuen Version auf wesentliche Probleme stoßen. Sie können kleinere Patches für Anwendungen als Einzeldatei bereitstellen und die Anzahl erforderlicher Rollbacks verringern.

  Application Link bietet Einsparungen bei der Bandbreite. Wenn Sie beispielsweise Microsoft Office 2007 Service Pack 1 haben und ohne Application Link auf Service Pack 2 aktualisieren möchten, müssten Sie mit der Bereitstellung eines neuen Office 2007 Service Pack 2-Pakets 1,5 GB an Daten pro Computer übertragen. Application Link überträgt ausschließlich die Updates und nicht das gesamte Paket auf die Computer.

Ansicht der Anwendung unter Verwendung von Application Link

Abbildung 4-1 zeigt die laufende Anwendung mit einer zusammengeführten Ansicht des Systems, der Basisanwendung und aller verknüpften Komponenten. Dateien, Registrierungsschlüssel, Dienste, COM-Objekte und Umgebungsvariablen von abhängigen Paketen sind in der Basisanwendung sichtbar.

Abbildung 4-1. Ansicht des Systems, der Basisanwendung und aller verknüpften Komponenten unter Verwendung von Application Link



Verknüpfen einer Basisanwendung mit Microsoft .NET Framework

Betrachten Sie dieses Workflow-Beispiel für das Verknüpfen einer Basisanwendung, `MyApp.exe`, mit einem getrennten Paket, das Microsoft .NET 2.0 Framework enthält. Stellen Sie sicher, dass die Kapselung der Basisanwendung Microsoft .NET 2.0 Framework nicht einschließt. Informationen über die Kapselung einer Anwendung finden Sie in Kapitel 2, „Kapselung von Anwendungen“, auf Seite 15.

Informationen über erforderliche und optionale Application Link-Parameter in der `Package.ini`-Datei finden Sie unter „Konfigurieren von abhängigen Anwendungen mit Application Link“ auf Seite 89.

So verknüpfen Sie eine Anwendung mit Microsoft .NET

1 Kapseln Sie die Installation von .NET 2.0 Framework.

   Während des Kapselungsprozesses müssen Sie mindestens einen Einstiegspunkt auswählen, auf den Benutzer Zugriff haben.

2 Benennen Sie die von ThinApp erzeugte `.exe`-Datei in eine `.dat`-Datei um.

   Durch diese Umbenennung wird verhindert, dass Benutzer die Anwendung aus Versehen ausführen.

   Der Name, den Sie für die `.dat`-Datei auswählen, ist unbedeutend, da die Benutzer die Datei nicht direkt ausführen. Verwenden Sie beispielsweise `dotnet.dat`.

3 Speichern Sie das .NET-Projekt unter `C:\Captures\dotnet`.

4 Kapseln Sie die Basisanwendung, indem Sie dasselbe physische System oder dieselbe virtuelle Maschine verwenden, auf dem/der .NET Framework bereits installiert ist.

5 Speichern Sie das Projekt unter `C:\Captures\MyApp`.

6 Öffnen Sie die `Package.ini`-Datei, die sich im gekapselten Anwendungsordner befindet, für die Basisanwendung.

7 Aktivieren Sie den `RequiredAppLinks`-Parameter für die Basisanwendung, indem Sie nach dem Eintrag `[BuildOptions]` folgende Zeile einfügen:

   ```
   RequiredAppLinks=dotnet.dat
   ```

   Application Link-Parameter müssen sich auf den primären Datencontainer der Anwendung, zu der Sie einen Link erstellen möchten, beziehen. Der Bezug auf `.exe`-Verknüpfungsdateien ist nicht möglich, da diese Dateien keine Anwendungen, Dateien oder Registrierungsschlüssel enthalten.

8 Erstellen Sie Pakete von .NET 2.0 und Basisanwendung erneut.

   a Doppelklicken Sie auf die Datei `build.bat` unter `C:\Captures\MyApp`.

   b Doppelklicken Sie auf die Datei `build.bat` unter `C:\Captures\dotnet`.

   Durch Ausführen dieser Batchdateien werden getrennte ThinApp-Pakete erstellt.

9 Stellen Sie diese Anwendungen auf einem Endbenutzer-Desktop unter `C:\Programme\MyApp` bereit.

   a Kopieren Sie `C:\Captures\MyApp\bin\MyApp.exe` in `\\<Endbenutzer_Desktop>\<Programme_Freigabe>\MyApp\MyApp.exe`.

   b Kopieren Sie `C:\Captures\dotnet\bin\cmd.exe` in `\\<Endbenutzer_Desktop>\<Programme_Freigabe>\MyApp\dotnet.dat`.

Einrichten verschachtelter Links mit Application Link

Mit dem Dienstprogramm Application Link unterstützt ThinApp verschachtelte Links. Wenn beispielsweise Microsoft Office einen Link zu einem Service Pack herstellt und das Service Pack einen Link zu einem Hotfix bietet, unterstützt ThinApp alle diese Abhängigkeiten.

Dieser Vorgang bezieht sich auf AppA, die AppB erfordert und AppB, die AppC erfordert. Für den Vorgang wird folgende Ordnerstruktur angenommen:

- `c:\AppFolder\AppA\AppA.exe`
- `c:\AppFolder\AppB\AppB.exe`
- `c:\AppFolder\AppC\AppC.exe`

Informationen über das Einrichten der erforderlichen und optionalen Application Link-Parameter bei diesem Verfahren finden Sie unter „Konfigurieren von abhängigen Anwendungen mit Application Link“ auf Seite 89.

So richten Sie verschachtelte Links ein

1 Kapseln Sie Anwendung A.

2 Legen Sie in der `Package.ini`-Datei Anwendung B als erforderlichen oder optionalen Anwendungslink fest.

   Fügen Sie beispielsweise `RequiredLinks=\AppFolder\AppB\AppB.exe` zur Datei hinzu.

3 Kapseln Sie Anwendung B.

4 Legen Sie in der `Package.ini`-Datei für Anwendung B die Anwendung C als erforderlichen oder optionalen Anwendungslink fest.

   Fügen Sie beispielsweise `RequiredLinks=\AppFolder\AppC\AppC.exe` zur Datei hinzu.

5 Kapseln Sie Anwendung C.

   Wenn Sie Anwendung A starten, kann das Programm auf die Dateien und Registrierungsschlüssel von Anwendung B zugreifen und Anwendung B kann auf die Dateien und Registrierungsschlüssel von Anwendung C zugreifen.

Beeinflussen der Isolationsmodi mit Application Link

ThinApp lädt mit dem Start der Anwendung eine Application Link-Ebene und führt Registrierungseinträge und Dateisystemverzeichnisse zusammen. Wenn ThinApp einen Registrierungsschlüssel oder Dateisystemverzeichnis ermittelt, der im bereits zusammengeführten Hauptpaket bzw. in der Hauptebene nicht vorhanden war, verwendet ThinApp den in der geladenen Ebene festgelegten Isolationsmodus. Ist der Registrierungsunterschlüssel oder das Dateisystemverzeichnis in dem Hauptpaket und einer Ebene vorhanden, die bereits zusammengeführt wurde, dann verwendet ThinApp den am meisten einschränkenden Isolationsmodus, der in einer der Ebenen oder im Hauptpaket festgelegt wurde. Die Reihenfolge der Isolationsmodi lautet „Full“, „WriteCopy“ und „Merged“, wobei für „Full“ die meisten und für „Merged“ die wenigsten Einschränkungen gelten.

PermittedGroups auf verknüpfte Pakete

Wenn Sie zwei Anwendungen verknüpfen und einen Wert für den `PermittedGroups`-Parameter festlegen, muss das Benutzerkonto, das zum Starten der Anwendung verwendet wird, mindestens zu einer der Active Directory-Gruppen für diesen Parameter in den `Package.ini`-Dateien beider Anwendungen gehören. Informationen über den `PermittedGroups`-Parameter finden Sie unter „Konfigurieren von Berechtigungen und Sicherheit“ auf Seite 71.

Sandbox-Änderungen für eigenständige und verknüpfte Pakete

Sandbox-Änderungen von verknüpften Paketen sind für die ausführbare Basisdatei nicht sichtbar. Sie können zum Beispiel Acrobat Reader als eigenständiges virtuelles Paket installieren und als mit der Basisanwendung Firefox verknüpftes Paket. Wenn Sie Acrobat Reader als eigenständige Anwendung starten, indem Sie das virtuelle Paket ausführen und Änderungen in den Voreinstellungen vornehmen, speichert ThinApp die Änderungen in der Sandbox für Acrobat Reader. Wenn Sie Firefox starten, kann Firefox diese Änderungen nicht erkennen, weil Firefox eine eigene Sandbox hat. Beim Öffnen einer `PDF`-Datei mit Firefox werden die Änderungen der Voreinstellungen, die in der eigenständigen Anwendung Acrobat Reader vorhanden sind, nicht angezeigt.

Importierreihenfolge für verknüpfte Pakete

ThinApp importiert verknüpfte Anwendungen entsprechend der Reihenfolge der Anwendungen im `RequiredAppLinks`- oder `OptionalAppLinks`-Parameter. Wenn einer der Parameter ein Platzhalterzeichen festlegt, durch das das Importieren von mehr als einer Datei ausgelöst wird, entscheidet die alphabetische Reihenfolge darüber, welches Paket zuerst importiert wird.

Der `OptionalAppLinks`-Parameter wird möglicherweise als `OptionalAppLinks=a.exe;b.exe;plugins\*.exe` angezeigt.

Bei Verwendung von `a.exe` und `b.exe` als Beispiel für ausführbare Dateien importiert ThinApp verknüpfte Pakete in der folgenden Reihenfolge:

- Basisanwendung
- `a.exe`
- `b.exe`
- Plug-Ins, geladen in alphabetischer Reihenfolge
- Verschachtelte Plug-Ins für `a.exe`
- Verschachtelte Plug-Ins für `b.exe`
- Verschachtelte Plug-Ins für den ersten Satz der Plug-Ins in dieser Liste

Informationen über verschachtelte Links finden Sie unter „Einrichten verschachtelter Links mit Application Link“ auf Seite 45.

Datei- und Registrierungskollisionen in verknüpften Paketen

Enthalten die Basisanwendung und ein mit der Basisanwendung verknüpftes Paket Datei- oder Registrierungseinträge am selben Speicherort, dann tritt eine Kollision auf.. Geschieht dies, entscheidet die Reihenfolge der Importe darüber, welches Paket Priorität hat. In solchen Fällen hat das zuletzt importierte Paket Priorität und die Datei- oder Registrierungsinhalte dieses Pakets sind für die laufenden Anwendungen sichtbar.

VBScript-Kollisionen in verknüpften Paketen

VBScript-Namenskollisionen können Skripte in anderen importierten Paketen an der Ausführung hindern. Wenn Sie Pakete mit Application Link verknüpfen und diese Pakete Skripte mit dem gleichen Namen besitzen, legt ThinApp die VBScripts von den verlinkten Paketen in einen gemeinsamen Datenpool. Bei Skripten mit dem gleichen Namen führt ThinApp das Skript von dem letzten importierten Paket aus und ignoriert das andere Skript.

Ein Basispaket enthält möglicherweise die Dateien `a.vbs` und `b.vbs` und ein abhängiges Paket enthält möglicherweise die Dateien `b.vbs` und `c.vbs`. Da zwischen den beiden `b.vbs`-Dateien eine Namenskollision besteht, überschreibt das VBScript in dem letzten importierten Paket, das in einem `RequiredAppLinks`- oder `OptionalAppLinks`-Parameter spezifiziert wurde, alle früher importierten Skripte mit dem gleichen Namen. In diesem Fall fasst ThinApp den Datenpool von vier `.vbs`-Dateien mit der Datei `a.vbs` vom Basispaket und den Dateien `b.vbs` und `c.vbs` von dem abhängigen Paket in einem gemeinsamen Datenpool zusammen.

VBScript-Funktionsreihenfolge in verknüpften Paketen

In einem Datenpool von VBScripts, die mit Application Link verknüpft sind, werden die Funktionen in den Hauptbereichen der Skripte in alphabetischer Reihenfolge der Skriptnamen ausgeführt. Die ThinApp-Rückruffunktionen in den Skripten werden in umgekehrter alphabetischer Reihenfolge der Skriptnamen im Datenpool ausgeführt.

Speichern mehrerer Versionen einer verknüpften Anwendung im selben Verzeichnis

Enthält das Verzeichnis ein verknüpftes Paket und Sie fügen eine aktualisierte Version des verknüpften Pakets zum selben Verzeichnis hinzu, dann erkennt das Dienstprogramm Application Link die aktualisierte Version und verwendet sie.

Verwenden von Application Sync für eine Basisanwendung und verknüpfte Pakete

Wenn Sie Application Link verwenden, um Pakete mit einem Basispaket zu verknüpfen, und wenn Sie das Basispaket starten, kann Application Sync nur das Basispaket aktualisieren. Wenn Sie zum Beispiel ein Microsoft Office 2007-Paket mit Application Sync-Einträgen in der `Package.ini`-Datei und ein Adobe Reader-Paket mit Application Sync-Einträgen in der `Package.ini`-Datei erstellen, und Application Link verwenden, um die beiden Pakete zu verknüpfen und dann Microsoft Office 2007 starten, aktualisiert Application Sync nur Microsoft Office 2007. Sie können sowohl Microsoft Office 2007 als auch Adobe Reader aktualisieren, indem Sie jede Anwendung einzeln starten.

Wenn Sie nicht alle Anwendungen aktualisieren und eine Basisanwendung mit einem nicht mehr gültigen Plug-In verknüpfen, kann die Basisanwendung das Plug-In dennoch laden und verwenden.

Anwendungs-Updates, die der Administrator auslöst

ThinApp bietet Administratoren die Dienstprogramme `AppSync.exe` und `sbmerge.exe`.

Das Dienstprogramm `AppSync.exe` erzwingt auf einem Clientcomputer ein Update von Application Sync.

Das Dienstprogramm `sbmerge.exe` erstellt inkrementelle Updates der Anwendungen. Ein Administrator kann das Dienstprogramm zum Beispiel verwenden, um ein Plug-In für Firefox zu integrieren oder um die Startseite einer Website auf eine neue Standardwebsite zu verweisen.

Erzwingen eines Updates von Application Sync auf Clientcomputern

Sie können mithilfe des Befehls `AppSync` ein Update von Application Sync auf einem Clientcomputer erzwingen. Möglicherweise möchten Sie ein Paket aktualisieren, das an einem Speicherort gespeichert wird, für den Standardbenutzer keinen Schreibzugriff haben. In dieser Situation können Sie nicht die Application Sync-Parameter verwenden, um beim Start einer Anwendung nach Updates zu suchen, da die Benutzer nicht über die erforderlichen Rechte zum Update des Pakets verfügen. Sie können ein tägliches Ausführen von `AppSync.exe` über ein Konto mit ausreichenden Berechtigungen planen. Die Application Sync-Parameter, wie `AppSyncUpdateFrequency`, in der `Package.ini`-Datei haben keine Auswirkungen auf den Befehl `AppSync`.

Verwenden Sie zum Erzwingen eines Application Sync-Updates den Befehl `AppSync <Anwendung_Sync_URL> <Ausführbarer_Dateipfad>`. Der Wert der URL ist derselbe wie die Application Sync-URL in der `Package.ini`-Datei und der Pfad der ausführbaren Datei ist der Pfad zur ausführbaren Datei, die aktualisiert werden soll.

Aktualisieren von Anwendungen mit Laufzeitänderungen

Das Dienstprogramm `sbmerge.exe` führt Laufzeitänderungen, die in der Sandbox der Anwendung aufgezeichnet werden, in ein ThinApp-Projekt zusammen. Ein typischer Workflow für dieses Dienstprogramm umfasst folgende Aufgaben:

- Kapseln einer Anwendung.
- Erstellen der Anwendung mit der `build.bat`-Datei.
- Ausführen einer gekapselten Anwendung und Anpassung der Einstellungen und der virtuellen Umgebung. ThinApp speichert die Änderungen in der Sandbox.
- Ausführen des Dienstprogramms `sbmerge.exe`, um die Registrierungs- und Dateisystemänderungen aus der Sandbox im ThinApp-Projekt zusammenzuführen.
- Erneutes Erstellen der gekapselten Anwendung mit der `build.bat`-Datei.
- Bereitstellen der aktualisierten Anwendung.

Sandbox-Änderungen mit Firefox zusammenführen

Dieses Verfahren für das Dienstprogramm `sbmerge.exe` verwendet Firefox 2.0.0.3 als Beispiel für eine gekapselte Anwendung.

So führen Sie Sandbox-Änderungen mit Firefox 2.0.0.3 zusammen

1 Kapseln Sie Firefox 2.0.0.3.

2 Doppelklicken Sie auf die `build.bat`-Datei im Ordner der gekapselten Anwendung, um das Anwendungspaket neu zu erstellen.

Ein Firefox 2.0.0.3-Pfad zur `build.bat`-Datei könnte beispielsweise `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3\build.bat` sein.

3 Erstellen Sie für den Sandbox-Speicherort ein `Thinstall`-Verzeichnis im `bin`-Verzeichnis.

4 Starten Sie Firefox und nehmen Sie eine Einstellungsänderung vor.

Ändern Sie beispielsweise die Startseite.

5 Navigieren Sie von der Befehlszeile zum Verzeichnis, in dem der ThinApp-Projektordner gespeichert ist.

Navigieren Sie zum Beispiel zu `C:\Programme\VMware\VMware ThinApp\Captures\Mozilla Firefox 2.0.0.3`.

6 Geben Sie in der Befehlszeile den Befehl `"C:\Programme\VMware\VMware ThinApp\sbmerge" Print` ein.

ThinApp druckt die Änderungen, die sich durch Verwenden der gekapselten Anwendung auf den Sandbox-Ordner auswirkten.

7 Geben Sie in der Befehlszeile den Befehl `"C:\Programme\VMware\VMware ThinApp\sbmerge" Apply` ein.

ThinApp leert den Thinstall-Ordner und führt die Sandbox-Änderungen mit der Anwendung zusammen.

sbmerge.exe-Befehle

Der Befehl `sbmerge.exe Print` zeigt die Sandbox-Änderungen an und nimmt weder an der Sandbox noch am ursprünglichen Projekt Änderungen vor.

Der Befehl `sbmerge.exe Apply` führt die Änderungen aus der Sandbox mit dem ursprünglichen Projekt zusammen. Dieser Befehl aktualisiert die Projektregistrierung und das Dateisystem gemäß den Änderungen und löscht das Sandbox-Verzeichnis.

Verwendung

```
"C:\Programme\VMware\VMware ThinApp\sbmerge" Print [<Optionale_Parameter>]
"C:\Programme\VMware\VMware ThinApp\sbmerge" Apply [<Optionale_Parameter>]
```

Optionale Parameter

Die optionalen `sbmerge.exe`-Parameter legen die Projekt- und Sandbox-Pfade fest und blockieren Statusmeldungen und das Zusammenführen der Sandbox-Dateien.

Parameter	Beschreibung
`–ProjectDir <Projekt_Pfad>`	Verwenden Sie den absoluten oder relativen Pfad zum Projektverzeichnis mithilfe des `–ProjectDir <Projekt_Pfad>`-Parameters, wenn Sie den `sbmerge.exe`-Befehl von einem Speicherort außerhalb des Projektordners aufrufen. Ein Beispielbefehl lautet `"C:\Programme\VMware\VMware ThinApp\sbmerge" Print –ProjectDir "C:\<Projektordner_Pfad>""`.
`–SandboxDir <Sandbox_Pfad>`	Wenn Sie eine gekapselte Anwendung starten, sucht sie in einem bestimmten Ordner nach der Sandbox. Siehe „Suchreihenfolge für die Sandbox“ auf Seite 101. Verwenden Sie den `–SandboxDir <Sandbox_Pfad>`-Parameter, um den Speicherort festzulegen, wenn Sie einen benutzerdefinierten Ordner für die Sandbox verwenden.
`–Quiet`	Blockiert das Drucken der Statusmeldungen.
`–Exclude <Ausgeschlossene_Datei>.ini`	Unterbindet das Zusammenführen bestimmter Dateien oder Registrierungseinträge aus der Sandbox. Sie können eine `.ini`-Datei festlegen, um den Inhalt, der ausgeschlossen werden soll, zu bestimmen. Diese Datei enthält getrennte Abschnitte, um Dateien festzulegen, wie `FileSystemIgnoreList` und `RegistryIgnoreList`. Das Dienstprogramm `sbmerge.exe` verwendet standardmäßig die `snapshot.ini`-Datei im Installationsordner von ThinApp, um bestimmte Inhalte von der Zusammenführung auszuschließen. Diese Option ermöglicht es, eine weitere `.ini`-Datei festzulegen, um zusätzliche Inhalte auszuschließen.

Automatische Anwendungs-Updates

Kann eine Anwendung automatisch aktualisiert werden, funktioniert ihr Update-Mechanismus mit ThinApp. Lädt die Anwendung das Update herunter und führt einen Installationsdienstprogramm oder ein Patch-Programm aus, geschieht dies innerhalb der virtuellen Umgebung und ThinApp speichert die Änderungen aus der Update-Software in der Sandbox. Startet die Anwendung neu, verwendet sie die Version der ausführbaren Datei in der Sandbox und nicht die ausführbare Datei aus dem ursprünglichen Paket.

Wenn Sie zum Beispiel Firefox 1.5 kapseln, fordert der Auto-Update-Mechanismus möglicherweise zum Upgrade auf Firefox 2.0 auf. Wenn Sie mit dem Upgrade fortfahren, lädt die Anwendung die Updates herunter, schreibt die Updates in die Sandbox und fordert zum Neustart der Anwendung auf. Führen Sie die gekapselte Anwendung erneut aus, wird Firefox 2.0 gestartet. Wenn Sie die Sandbox löschen, kehrt Firefox zu Version 1.5 zurück.

Verwenden Sie das Dienstprogramm `sbmerge.exe`, um Änderungen zusammenzuführen, die ein Auto-Update-Mechanismus mit dem ursprünglichen Paket vornimmt, damit eine aktualisierte ausführbare Datei erstellt wird. Siehe „Anwendungs-Updates, die der Administrator auslöst“ auf Seite 48.

ANMERKUNG Deaktivieren Sie die Auto-Update-Funktionen der Anwendung, wenn Sie das Dienstprogramm Application Sync verwenden, um Anwendungs-Updates vorzunehmen. Siehe „Verwendung von Application Sync in einer verwalteten oder nicht verwaltete Umgebung“ auf Seite 41.

Dynamische Updates ohne Administratorrechte

Sie können Anwendungen dynamisch aktualisieren, ohne dass hierzu Administratorrechte erforderlich sind. Zum Beispiel müssen .NET-basierte Anwendungen, die als Teil ihres Update-Vorgangs neue DLL-Dateien aus dem Internet herunterladen, die `ngen.exe`-Datei ausführen, um Native-Image-Assemblies für den Startvorgang zu generieren. Unter normalen Umständen schreibt die `ngen.exe`-Datei in `HKLM` und `C:\WINDOWS`, auf die beiden nur mit Administratorkonten zugegriffen werden kann. Mit ThinApp kann die `ngen.exe`-Datei Native-Image-Assemblies auf Gastbenutzerkonten installieren; die Änderungen werden jedoch in einem benutzerspezifischen Verzeichnis gespeichert.

Sie können das Paket auf einem zentralen Computer aktualisieren und die Änderungen als neue gekapselte ausführbare Datei auf Clientcomputer oder zentrale Netzwerkfreigaben verschieben. Verwenden Sie eine der folgenden Optionen, um Updates anzuwenden:

- Im Verlauf des Setup Capture-Prozesses.
- Innerhalb der virtuellen Umgebung.

  Anwendungen mit Auto-Update-Funktionen können aktualisiert werden. Ist das Update eine `patch.exe`-Datei, kann das Patch-Programm in der virtuellen Umgebung und von einem `cmd.exe`-Dateieinstiegspunkt aus ausgeführt werden. Änderungen treten während automatischer oder manueller Updates in der Sandbox auf, um durch Löschen der Sandbox die Rückkehr zur ursprünglichen Version zu ermöglichen.

  Wenn Sie in der virtuellen Umgebung Patches auf einen zentralen Computer für die Paketierung anwenden, können Sie das Dienstprogramm `sbmerge.exe` verwenden, um die Sandbox-Änderungen, die durch das Update in der Anwendung vorgenommen wurden, zusammenzuführen. Siehe „Anwendungs-Updates, die der Administrator auslöst“ auf Seite 48.
- Im gekapselten Projekt.

  Zum Update kleiner Sätze an Dateien oder Registrierungsschlüsseln können Sie die Dateien im gekapselten Projekt ersetzen. Dieser Ansatz ist für Softwareentwickler nützlich, die ThinApp-Builds in ihren Workflow integrieren.

Aktualisieren von laufenden Anwendungen auf einer Netzwerkfreigabe

ThinApp ermöglicht Ihnen das Upgrade oder das Rollback einer Anwendung, die auf einer Netzwerkfreigabe für mehrere Benutzer ausgeführt wird. Das Upgrade beginnt, wenn der Benutzer die Anwendung beendet und dann erneut startet. In Terminalserver-Umgebungen können mehrere Benutzer während des Übergangszeitraums verschiedene Versionen gleichzeitig ausführen.

Dateisperren

Durch Starten einer Anwendung wird das ausführbare Dateipaket gesperrt. Sie können die Anwendung weder ersetzen, noch löschen oder verschieben. Diese Dateisperre gewährleistet, dass jedem Computer oder Benutzer, der auf eine bestimmte Version einer Anwendung zugreift, diese Version weiterhin zur Verfügung steht, solange die Anwendungsprozesse und Unterprozesse ausgeführt werden.

Wenn Sie eine Anwendung für viele Benutzer an einem zentralen Speicherort speichern, verhindert diese Dateisperre, dass Administratoren eine paketierte ausführbare Datei durch eine neue Version ersetzen, bevor sämtliche Benutzer die Anwendung beendet und ihre Sperren aufgehoben haben.

Upgrade einer laufenden Anwendung

Sie können eine neue Version einer Anwendung mit einer höheren Dateinamenerweiterung, wie .1 oder .2, in ein bestehendes Bereitstellungsverzeichnis kopieren. Bei diesem Vorgang wird Firefox als Beispielanwendung verwendet.

Verknüpfungen müssen nicht aktualisiert werden.

So aktualisieren Sie eine laufende Anwendung

1 Stellen Sie die ursprüngliche Version der Anwendung, wie `Firefox.exe`, bereit.

2 Kopieren Sie die Anwendung auf eine zentrale Freigabe unter `\\<Server>\<Freigabe>\Firefox.exe`.

  Ein Beispielspeicherort ist `C:\Programme\Firefox\Firefox.exe`.

3 Erstellen Sie eine Desktop- oder **Start**menüverknüpfung zum Benutzer-Desktop, die unter `\\<Server>\<Freigabe>\Firefox.exe` auf den Speicherort einer gemeinsam genutzten Datei zeigt.

  Annahme: Zwei Benutzer starten `Firefox.exe` und sperren die Anwendung.

4 Kopieren Sie die aktualisierte Version der `Firefox.exe`-Datei auf die zentrale Freigabe unter `\\<Server>\<Freigabe>\Firefox.1`.

   Sind Sie ein neuer Benutzer, startet ThinApp die Anwendung mit den neuen Paketdaten von `Firefox.1`. Sind Sie ein Benutzer, der mit der ursprünglichen Version arbeitet, können Sie die neue Version sehen, nachdem Sie die Anwendung beendet und die Anwendung neu gestartet haben.

5 Wenn Sie ein neueres Update von Firefox bereitstellen müssen, speichern Sie sie mit einer höheren Versionsnummer am Ende in dasselbe Verzeichnis.

6 Kopieren Sie Version 2.0 von `Firefox.exe` auf einen zentralen, gemeinsam genutzten Server unter `\\<Server>\<Freigabe>\Firefox.2`

Nachdem die Sperre von `Firefox.1` aufgehoben wurde, können Sie die Version löschen, aber die `Firefox.exe`-Datei sollte bestehen bleiben, da die Benutzerverknüpfungen weiterhin auf diese Datei zeigen. ThinApp benutzt stets den Dateinamen mit der höchsten Versionsnummer. Wenn Sie ein Rollback auf eine frühere Version vornehmen müssen und die neueste Version immer noch gesperrt ist, kopieren Sie die alte Version, sodass sie die höchste Versionsnummer trägt.

Sandbox-Überlegungen für aktualisierte Anwendungen

Wenn Sie eine Anwendung aktualisieren, können Sie steuern, ob die Benutzer weiterhin ihre früheren Einstellungen verwenden können, indem Sie den Sandbox-Namen in der `Package.ini`-Datei konsistent halten. Sie können Benutzer daran hindern, eine ältere Sandbox mit einer aktualisierten Anwendung zu verwenden, indem Sie die aktualisierte Anwendung mit einem neuen Namen für die Sandbox verpacken. Beim ersten Start der aktualisierten Anwendung wird die Sandbox mit dem neuen Namen erstellt.

ThinApp – Überwachung und Problembehandlung 5

Verwenden Sie Protokoll-Monitor zum Generieren von Trace-Dateien und zur Problembehandlung innerhalb der ThinApp-Umgebung. Protokoll-Monitor ist ausschließlich mit einer gekapselten Anwendung, die dieselbe Version von ThinApp verwendet, kompatibel.

Dieser Abschnitt umfasst die folgenden Themen:

- „Bereitstellen von Informationen für den technischen Support“ auf Seite 53
- „Protokoll-Monitor-Vorgänge“ auf Seite 54
- „Problembehandlung bei bestimmten Anwendungen“ auf Seite 61

Bereitstellen von Informationen für den technischen Support

Zur Problembehandlung innerhalb einer ThinApp-Umgebung benötigt der technische Support von VMware folgende Angaben von Ihnen:

- Eine schrittweise Schilderung der Vorgänge, die Sie vornahmen, als das Problem auftrat.
- Informationen über die Hostkonfiguration. Geben Sie das Windows-Betriebssystem an, die Verwendung von Terminal Server oder Citrix Xenapp und alle erforderlichen Programme, die auf dem nativen System installiert sind.
- Kopien der Trace-Dateien von Protokoll-Monitor. Siehe „Protokoll-Monitor-Vorgänge“ auf Seite 54.
- Eine genaue Kopie des Kapselungsordners und seines gesamten Inhalts. Beziehen Sie die kompilierten ausführbaren Dateien aus dem `/bin`-Unterordner nicht mit ein.
- Eine Beschreibung des erwarteten und des tatsächlichen Verhaltens der Anwendung.
- (Optional) Kopien der Anwendungen, die Sie gekapselt haben. Schließen Sie die Server-Komponentenkonfiguration für Oracle Server oder Active Directory mit ein.
- (Optional) Native oder physische Dateien oder Einstellungen für Registrierungsschlüssel, die für das Problem bedeutsam sein könnten.
- (Optional) Systemdienste oder erforderliche Gerätetreiber.
- (Optional) Die virtuelle Maschine, die den Fehler nachbildet. Der VMware-Support benötigt sie möglicherweise, wenn die Support-Kontaktperson nicht in der Lage ist, das Problem nachzustellen.
- (Optional) Eine oder mehrere WebEx-Sitzungen, um das Debugging in Ihrer Systemumgebung zu vereinfachen.

Protokoll-Monitor-Vorgänge

Protokoll-Monitor erfasst detailliert und chronologisch die Aktivitäten ausführbarer, von der gekapselten Anwendung gestarteter Dateien. Protokoll-Monitor fängt Protokollnamen, Adressen, Parameter und Rückgabewerte für jeden Funktionsaufruf der ausführbaren Zieldateien oder DLLs ab und protokolliert sie. Protokoll-Monitor erfasst folgende Aktivitäten:

- Win32 API-Aufrufe von Anwendungen, die im virtuellen Betriebssystem von ThinApp ausgeführt werden.
- Potenzielle Fehler, Ausnahmen und Sicherheitsereignisse innerhalb der Anwendung.
- Alle DLLs, die von der Anwendung und den Adressbereichen geladen werden.

Die generierten Protokolldateien können größer als 100 MB sein, je nachdem, wie lange die Anwendung mit Protokoll-Monitor ausgeführt wird und wie ausgelastet eine Anwendung ist. Der einzige Grund, warum Protokoll-Monitor für eine Anwendung ausgeführt wird, ist die Erfassung der Trace-Dateien. Trace-Dateien sind wesentlich für die Problembehandlung, da innerhalb der Trace-Datei mehrere Einträge analysiert und korreliert werden.

Problembehandlung mit Protokoll-Monitor

Sie können Protokoll-Monitor für die grundlegende Problembehandlung einsetzen.

Problembehandlung mithilfe der ThinApp-Protokolle

1 Beenden Sie die gekapselte Anwendung vor der Untersuchung.

2 Wählen Sie auf dem Computer, auf dem Sie die Anwendung gekapselt haben, **Start** > **Alle Programme** > **VMware** > **ThinApp Log Monitor**.

Um Protokoll-Monitor von einem Arbeitscomputer aus zu starten, kopieren Sie die Dateien `log_monitor.exe`, `logging.dll` und `Setup Capture.exe` von `C:\Programme\VMware\VMware ThinApp` auf den Arbeitscomputer und doppelklicken Sie auf die Datei `log_monitor.exe`.

3 Starten Sie die gekapselte Anwendung.

Mit dem Start der Anwendung wird in der Protokoll-Monitor-Liste ein neuer Eintrag angezeigt. Protokoll-Monitor zeigt für jede neue Trace-Datei einen neuen Eintrag an. Eine einzelne Datei muss nicht unbedingt an einen einzelnen Vorgang gekoppelt sein.

4 Beenden Sie die Anwendung, sobald ein Fehler auftritt.

5 Generieren Sie Protokolle für jede Trace-Datei, die Sie untersuchen möchten.

a Wählen Sie die `.trace`-Datei aus der Liste aus.

b Klicken Sie auf **Trace-Textbericht generieren (Generate text trace report)**.

Vom übergeordneten Prozess generierte untergeordnete Prozesse sind im gleichen Protokoll gespeichert. Mehrere, unabhängige Prozesse sind nicht im selben Protokoll gespeichert.

ThinApp generiert eine `.trace`-Datei. Protokoll-Monitor konvertiert die binäre `.trace`-Datei ein eine `.txt`-Datei.

6 (Optional) Öffnen Sie die `.txt`-Datei in einem Text-Editor und überprüfen Sie die Informationen. Unter bestimmten Umständen ist die `.txt`-Datei zu groß, um mit dem Text-Editor geöffnet zu werden.

7 Zippen Sie die `.txt`-Dateien und senden Sie die Dateien an VMware-Support.

Ausführen von erweiterten Protokoll-Monitor-Vorgängen

Zu den erweiterten Protokoll-Monitor-Vorgängen gehört das Beenden von Anwendungen oder das Löschen von Trace-Dateien. Ist eine Anwendung beschäftigt oder weist sie bei einer bestimmten Aktion nur eine verlangsamte Leistung auf, können Sie die Vorgänge **Anhalten (Suspend)** und **Fortsetzen (Resume)** ausführen, um die Protokolle eines bestimmten Zeitraums zu erfassen. Die resultierende Protokolldatei ist kleiner als die typische Protokolldatei und einfacher zu analysieren. Auch bei der Verwendung der Vorgänge **Anhalten (Suspend)** und **Fortsetzen (Resume)** kann die ursprüngliche Ursache für einen Fehler außerhalb des Zeitfensters liegen. Die Vorgänge **Anhalten (Suspend)** und **Fortsetzen (Resume)** sind global und wirken sich auf sämtliche Anwendungen aus.

Um weitere Informationen über die Verwendung dieser Optionen zu erhalten, wenden Sie sich bitte an VMware-Support.

So führen Sie erweiterte Protokoll-Monitor-Vorgänge aus

1 Beenden Sie die gekapselte Anwendung vor der Untersuchung.

2 Wählen Sie auf dem Computer, auf dem Sie die Anwendung gekapselt haben, **Start** > **Alle Programme** > **VMware** > **ThinApp Log Monitor**.

   Um Protokoll-Monitor von einem Arbeitscomputer aus zu starten, kopieren Sie die Dateien `log_monitor.exe`, `logging.dll` und `Setup Capture.exe` von `C:\Programme\VMware\VMware ThinApp` auf den Arbeitscomputer und doppelklicken Sie auf die Datei `log_monitor.exe`.

3 (Optional) Erfassen Sie die Protokolle eines bestimmten Zeitraums, um einen zeitlich genau festgelegten Fehler zu suchen.

   a Aktivieren Sie das Kontrollkästchen **Anhalten (Suspend)**.

   b Starten Sie die gekapselte Anwendung und führen Sie sie bis zum Punkt aus, an dem der Fehler auftritt oder die Leistungsminderung beginnt.

   c Deaktivieren Sie im Protokoll-Monitor das Kontrollkästchen **Anhalten (Suspend)**, um den Protokollierungsvorgang wiederaufzunehmen.

      Sie können das Verhalten der Anwendung prüfen, um das Problem zu isolieren.

   d Aktivieren Sie das Kontrollkästchen **Anhalten (Suspend)**, um den Protokollierungsvorgang anzuhalten.

4 (Optional) Wählen Sie in der Trace-Dateiliste die zu löschende Datei und klicken Sie auf **Datei löschen (Delete File)**.

5 (Optional) Klicken Sie auf **Abbrechen (Kill App)**, um einen laufenden Prozess anzuhalten.

6 (Optional) Aktivieren Sie das Kontrollkästchen **Komprimieren (Compress)**, um die Größe einer Trace-Datei zu verringern.

   Dieser Vorgang vermindert die Leistung der Anwendung.

7 (Optional) Generieren Sie einen Trace-Dateibericht.

   a Wählen Sie in der Dateiliste eine Trace-Datei aus, geben Sie einen Trace-Dateinamen ein oder klicken Sie auf **Durchsuchen (Browse)**, um eine Trace-Datei auf Ihrem System auszuwählen.

   b (Optional) Geben Sie den Namen des Ausgabeberichts ein oder verändern Sie den Namen des Ausgabeberichts.

   c Klicken Sie auf **Trace-Textbericht generieren (Generate text trace report)**, um einen Bericht zu erstellen.

      Sie können die Datei mit einem Text-Editor anzeigen, der Zeilenumbrüche nach dem UNIX-System unterstützt.

Ermitteln von Fehlern

Die ThinApp-Protokollierung stellt eine große Menge an Informationen bereit. Die folgenden Tipps sollen fortgeschrittene Anwender bei der Fehleruntersuchung unterstützen:

- Lesen Sie den Abschnitt `Potential Errors Detected (Erkannte potenzielle Fehler)` der `.txt`-Trace-Datei.

  Die Einträge zeigen möglicherweise keine Fehler an. ThinApp listet jeden Win32 API-Aufruf auf, bei dem sich der Windows-Fehlercode verändert hat.

- Prüfen Sie die von den Anwendungen generierten Ausnahmen.

  Ausnahmen können auf Fehler hinweisen. Zu den Ausnahmetypen gehören C++ und .NET. Die Trace-Datei zeichnet den Ausnahmetyp auf und die DLL, die die Ausnahme generiert. Wenn eine Anwendung, wie .NET oder Java, eine Ausnahme aus einem selbstgenerierenden Code erzeugt, zeigt die Trace-Datei ein unbekanntes Modul an.

  Das folgende Beispiel ist ein `.trace`-Eintrag für eine Ausnahme.

  ```
  *** Exception EXCEPTION_ACCESS_VIOLATION on read of 0x10 from unknown_module:0x7c9105f8
  ```

  Wenn Sie auf eine Ausnahme stoßen, überprüfen Sie den Anfangsteil der Trace-Datei auf die Ursache der Ausnahme. Ignorieren Sie Gleitkommaausnahmen, die Virtual Basic 6-Anwendungen bei normaler Verwendung generieren.

- Betrachen Sie die untergeordneten Prozesse.

  Protokoll-Monitor erstellt für jeden Prozess eine `.trace`-Datei. Startet eine Anwendung mehrere untergeordnete Prozesse, müssen Sie ermitteln, welcher Prozess den Fehler verursacht. Unter bestimmten Umständen, beispielsweise bei prozessunabhängigem COM, verwendet eine übergeordnete Anwendung COM, um einen untergeordneten Prozess zu starten, führt eine Funktion remote aus und fährt mit der Ausführung von Funktionen fort.

- Wenn Sie Anwendungen von einer Netzwerkfreigabe aus starten, die zwei Prozesse generiert, ignorieren Sie den ersten Prozess.

  ThinApp begegnet der herabgesetzten Leistung von Symantec Antivirus-Anwendungen durch erneutes Starten der Prozesse.

- Suchen Sie die in den Dialogfeldern angezeigte Fehlermeldung.

  Einige Anwendungen rufen die Funktion MessageBox Win32 API auf, um unerwartete Fehler während der Laufzeit anzuzeigen. Sie können in einer Trace-Datei nach einem `MessageBox` suchen oder nach dem Inhalt der Zeichenfolge, die in der Fehlermeldung angezeigt wird, und feststellen, welche Anwendung zuletzt ausgeführt wurde, bevor das Dialogfeld angezeigt wurde.

- Engen Sie die Auswahl an Aufrufen von einer bestimmten DLL und einem bestimmten Thread ein.

  Das Protokollformat legt die DLL und den Thread fest, die einen Aufruf auslösen. Die Aufrufe von System-DLLs können häufig ignoriert werden.

Protokollformat

Eine Trace-Datei enthält folgende Abschnitte:

- Systemkonfiguration

  Dieser Abschnitt enthält Informationen über das Betriebssystem, die Laufwerke, die installierten Programme, die Umgebungsvariablen, die Prozessliste, die Dienste und Treiber.

  Die Informationen beginnen mit der Zeichenfolge `Dump started on (Abbildsicherung gestartet um)` und enden mit der Zeichenfolge `Dump ended on (Abbildsicherung beendet um)`.

- Kopfzeile

In diesem Abschnitt werden Kontextinformationen angezeigt, zum Beispiel über den durch Protokoll-Monitor nachverfolgten Prozess. Einige der angezeigten Attribute zeigen die Anmeldeoptionen, Adressbereiche beim Laden der Laufzeit des Betriebssystems und die Makrozuordnung zu bestehenden Systempfaden.

ThinApp markiert den Beginn des Kopfzeilenabschnitts mit der Sequenzziffer 000001. Unter normalen Umständen markiert ThinApp das Ende dieses Abschnitts mit einer Meldung über das Dienstprogramm Application Sync.

- Textkörper

  Dieser Abschnitt enthält die Trace-Aktivität beim Starten und Ausführen von Vorgängen durch die Anwendung. Jede Zeile stellt einen Funktionsaufruf dar, den die ausführbaren Zieldateien oder eine der DLLs vornehmen.

  Der Abschnitt beginnt mit dem Eintrag `New Modules detected in memory (Im Arbeitsspeicher erkannte neue Module)`, gefolgt von der Modulliste `SYSTEM_LOADED`. Der Abschnitt endet mit dem Eintrag `Modules Loaded (Geladene Module)`.

- Zusammenfassung

  Dieser Abschnitt enthält Module, die von der gekapselten Anwendung geladen werden, potenzielle Fehler und ein Profil der 150 langsamsten Aufrufe.

  Der Abschnitt beginnt mit der Meldung `Modules Loaded (Geladene Module)`.

Allgemeines Format der API-Protokollmeldungen

Die folgende Meldung ist ein Beispiel für das Format von API-Aufrufen.

```
000257 0a88 mydll.dll :4ad0576d–>kernel32.dll:7c81b1f0 SetConsoleMode (IN HANDLE
hConsoleHandle=7h, IN DWORD dwMode=3h)
000258 0a88 mydll.dll :4ad0576d<–kernel32.dll:7c81b1f0 SetConsoleMode –>BOOL=1h ()
```

Dieses Beispiel enthält folgende Einträge:

- `000257` steht für die Protokolleintragsnummer. Jeder Protokolleintrag besitzt eine eigene Nummer.
- `0a88` steht für die Kennung des aktuell ausgeführten Threads. Hat die Anwendung nur einen Thread, bleibt diese Nummer unverändert. Zeichnen zwei oder mehrere Threads Daten in der Protokolldatei auf, kann man die Thread-Kennung verwenden, um die zu einem Thread gehörenden aufeinanderfolgenden Aktionen nachzuverfolgen, da ThinApp Protokolleinträge in der Reihenfolge ihres Auftretens verzeichnet.
- `mydll.dll` steht für die DLL, die den API-Aufruf vornimmt.
- `4ad0576d` steht für die Rücksprungadresse für den API-Aufruf, den `mydll.dll` vornimmt. Unter normalen Umständen ist die Rücksprungadresse die Adresse im Code, bei der der Aufruf durchgeführt wurde.
- `–>` zeigt den Beginn des Funktionsaufrufs an. Für den Protokolleintrag der begonnenen Ausführung des Funktionsaufrufs zeigt ThinApp die Eingabeparameter an. Diese Parameter sind in- und in-/out-Parameter.
- `<–` zeigt die Rückkehr vom Funktionsaufruf zur ursprünglichen Einsprungstelle an. Für Protokolleinträge zur Funktionsrückkehr zeigt ThinApp die Ausgabeparameter an. Diese Parameter sind out- und in/out-Parameter.
- `kernel32.dll` steht für die DLL, von der der API-Aufruf ausgeführt wird.
- `7c81b1f0` steht für die Adresse der API-Funktion innerhalb der kernel32.dll, von der der Aufruf ausgeführt wird. Indem man `kernel32.dll` an der `7c81b1f0`-Adresse disassembliert, kann man den Code für die Funktion SetConsoleMode anzeigen.
- `–>BOOL=1h` zeigt an, dass die API-Funktion den Wert 1 zurückgibt und der Rückgabewert vom Typ BOOL ist.

Informationen zum Anwendungsstart

Die folgenden Einträge bieten grundlegende Informationen über die Anwendung, beispielsweise den Modulnamen und die Prozess-ID (PID), sowie über Protokoll-Monitor, beispielsweise die Version und die Optionen.

```
000001 0a88 Logging started for Module=C:\test\cmd_test\bin\cmd.exe
Using archive=
PID=0xec
CommandLine = cmd
000002 0a88 Logging options: CAP_LEVEL=9 MAX_CAP_ARY=25 MAX_CAP_STR=150
MAX_NEST=100
VERSION=3.090

000003 0a88 System Current Directory = C:\test\cmd_test\bin Virtual Current Directory =
C:\test\cmd_test\bin

000004 0a88 |start_env_var| =::=::\
000005 0a88 |start_env_var| =C:=C:\test\cmd_test\bin
000006 0a88 |start_env_var| =ExitCode=00000000
000007 0a88 |start_env_var| ALLUSERSPROFILE=C:\Dokumente und Einstellungen\All Users.WINDOWS
...
...
...
```

Liste der während der Laufzeit in den Arbeitsspeicher geladenen DLLs

Der Abschnitt `Modules loaded (Geladene Module)` findet sich nahe dem Ende der Protokolldatei und beschreibt die DLLs, die während der Laufzeit in den Arbeitsspeicher geladen werden, sowie die DLL-Adressen. Die Informationen zeigen an, ob Windows oder ThinApp die DLLs lädt.

Dieses Beispiel enthält eine Zusammenfassung der Länge der längsten Aufrufe sowie die folgenden Einträge:

- `SYSTEM_LOADED` zeigt an, dass Windows die DLL lädt. Die Datei muss auf der Festplatte vorhanden sein.
- `MEMORY_MAPPED_ANON` zeigt an, dass ThinApp die DLL lädt. ThinApp kann die Datei vom virtuellen Dateisystem laden.
- `46800000-46873fff` zeigt den Adressbereich im virtuellen Arbeitsspeicher an, in dem sich die DLL befindet.
- `PRELOADED_BY_SYSTEM` und `PRELOADED_MAP` sind Duplikateinträge, die sich auf den Arbeitsspeicheradressbereich beziehen, in dem die ausführbare Image-Datei im Arbeitsspeicher abgebildet wird.

```
---Modules loaded --
PRELOADED_MAP 00400000-00452fff, C:\Programme\Adobe\Reader
8.0\Reader\AcroRd32.exe
PRELOADED_BY_SYSTEM 00400000-00452fff, C:\Programme\Adobe\Reader
8.0\Reader\AcroRd32.exe
SYSTEM_LOADED 00400000-00452fff, C:\Test\AcroRd32.exe
MEMORY_MAPPED_ANON 013b0000-020affff, C:\Programme\Adobe\Reader
8.0\Reader\AcroRd32.dll

----Timing Report: list of slowest 150 objects profiled ---

8255572220 total cycles (2955.56 ms): |sprof| thinapp_LoadLibrary2

765380728 cycles (274.01 ms) on log entry 21753
428701805 cycles (153.48 ms) on log entry 191955
410404281 cycles (146.93 ms) on log entry 193969
.
.
... 438 total calls
7847975891 total cycles (2809.64 ms): |sprof| ts_load_internal_module
764794646 cycles (273.80 ms) on log entry 21753
426837866 cycles (152.81 ms) on log entry 191955
408570540 cycles (146.27 ms) on log entry 193969
```

```
.
.
... 94 total calls
4451728477 total cycles (1593.76 ms): |sprof| ts_lookup_imports
544327945 cycles (194.87 ms) on log entry 21758
385149968 cycles (137.89 ms) on log entry 193970
187246661 cycles (67.04 ms) on log entry 190210
.
.
... 34 total calls
1099873523 total cycles (393.76 ms): |sprof| new_thread_start
561664565 cycles (201.08 ms) on log entry 151922
531551734 cycles (190.30 ms) on log entry 152733
1619002 cycles (0.58 ms) on log entry 72875
```

Potenzielle Fehler

Der Abschnitt `Erkannte potenzielle Fehler (Potential Errors Detected)` zeigt die Protokolleinträge an, deren Zeichenfolgen Probleme mit drei Sternen (***) aufweisen. Informationen zur Interpretation dieses Abschnitts erhalten Sie unter „Ermitteln von Fehlern“ auf Seite 56.

```
----Potential Errors Detected ---

006425 0000075c        LoadLibraryExW 'C:\Programme\Adobe\Reader
8.0\Reader\Microsoft.Windows.Common-Controls.DLL' flags=2 -> 0 (failed ***)
006427 0000075c        LoadLibraryExW 'C:\Programme\Adobe\Reader
8.0\Reader\Microsoft.Windows.Common-Controls\Microsoft.Windows.Common-Controls.DLL' flags=2
-> 0 (failed ***)
006428 0000089c nview.dll :1005b94b<-kernel32.dll:7c80ae4b *** LoadLibraryW -
>HMODULE=7c800000h () *** GetLastError() returns 2 [0]: The system cannot find the file
specified.
007062 0000075c        LoadLibraryExW 'C:\Programme\Adobe\Reader
8.0\Reader\en-US\Microsoft.Windows.Common-Controls.DLL' flags=2 -> 0 (failed ***)
010649 0000075c        LoadLibraryExW 'C:\Programme\Adobe\Reader
8.0\Reader\en-US\Microsoft.Windows.Common-Controls\Microsoft.Windows.Common-Controls.DLL'
flags=2 -> 0 (failed ***)
019127 0000075c MSVCR80.dll :781348cc<-msvcrt.dll :77c10396 *** GetEnvironmentVariableA -
>DWORD=0h (OUT LPSTR lpBuffer=*0h <bad ptr>) *** GetLastError() returns 203 [0]: The system
could not find the environment option that was entered.
019133 0000075c MSVCR80.dll :78133003<-nview.dll :1000058c *** GetProcAddress -
>FARPROC=*0h () *** GetLastError() returns 127 [203]: The specified procedure could not be found.
019435 0000075c MSVCR80.dll :78136e08<-dbghelp.dll :59a60360 *** Getfile type ->DWORD=0h ()
*** GetLastError() returns 6 [0]: The handle is invalid.
019500 0000075c MSVCR80.dll :78134481<-nview.dll :1000058c *** GetProcAddress -
>FARPROC=*0h () *** GetLastError() returns 127 [0]: The specified procedure could not be found.
019530 0000075c MSVCR80.dll :78131dcd<-dbghelp.dll :59a603a1 *** GetModuleHandleA -
>HMODULE=0h () *** GetLastError() returns 126 [0]: The specified module could not be found.
```

Beispiel für die Problembehandlung beim Dienstprogramm cmd.exe

Im Problembehebungsbeispiel verpackt ThinApp das Dienstprogramm `cmd.exe` bei aktivierter Protokollierung. Dieses Beispiel zeigt, wie Sie durch Ausführen eines ungültigen Befehls das fehlerhafte Verhalten einer Anwendung simulieren können. Wenn das Dienstprogramm `cmd.exe` dazu aufgefordert wird, den Befehl `foobar` auszuführen, generiert das Dienstprogramm die Meldung `foobar is not recognized as an internal or external command (foobar wird nicht als interner oder externer Befehl erkannt)`. Sie können die Trace-Datei und den Abschnitt `Erkannte potenzielle Fehler (Potential Errors Detected)` überprüfen, um die API-Funktionen zu suchen, die den Code `GetLastError` veränderten.

Das Beispiel zeigt die Pfade `C:\test\cmd_test\bin\foobar.*`, `C:\WINDOWS\system32\foobar.*` und `C:\WINDOWS\foobar` als Speicherorte, an denen das Dienstprogramm `cmd.exe` nach dem Befehl `foobar` sucht.

Das Beispiel zeigt die Pfade `%drive_C%\test\cmd_test\bin`, `%SystemSystem%\foobar` und `%SystemRoot%\foobar` als Speicherorte im virtuellen Dateisystem, die ThinApp durchsucht.

```
----Potential Errors Detected ---
*** Unable to determine if any services need to be auto-started, error 2 (Es konnte nicht
ermittelt werden, ob Dienste automatisch gestartet werden müssen, Fehler 2)
001550 *** FindFirstFileW 'C:\test\cmd_test\bin\foobar.*' -> INVALID_HANDLE_VALUE *** failed
[system probe C:\test\cmd_test\bin\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW ->HANDLE=ffffffffh .. *** GetLastError() returns 2 [203]: The system cannot
find the file specified.
*** FindFirstFileW 'C:\test\cmd_test\bin\foobar' -> INVALID_HANDLE_VALUE *** failed [FS
missing in view 0][fs entry not found %drive_C%\test\cmd_test\bin\foobar][fs entry not found
%drive_C%\test\cmd_test\bin]
*** FindFirstFileW 'C:\WINDOWS\system32\foobar.*' -> INVALID_HANDLE_VALUE *** failed [system
probe C:\WINDOWS\system32\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW 'C:\WINDOWS\system32\foobar' -> INVALID_HANDLE_VALUE *** failed [FS missing
in view 0][fs entry not found %SystemSystem%\foobar]
*** FindFirstFileW 'C:\WINDOWS\foobar.*' -> INVALID_HANDLE_VALUE *** failed [system probe
C:\WINDOWS\foobar.* -> ffffffffh][no virtual or system matches]
*** FindFirstFileW 'C:\WINDOWS\foobar' -> INVALID_HANDLE_VALUE *** failed [FS missing in view
0][fs entry not found %SystemRoot%\foobar]
```

Ausführen von erweiterten Untersuchungen von cmd.exe-Protokolleinträgen

Eine gründlichere Untersuchung eines Eintrags aus dem Abschnitt `Potential Errors Detected (Erkannte potenzielle Fehler)` in einer Trace-Datei kann die Suche in der gesamten Trace-Datei von Protokoll-Monitor nach dem betreffenden Eintrag umfassen sowie die Prüfung der Systemaufrufe und der Bedingungen, die zum potenziellen Fehler führten.

Der folgende Eintrag für das Dienstprogramm `cmd.exe` im Abschnitt `Potenzielle Fehler (Potential Errors)` könnte beispielsweise eine gründlichere Untersuchung durch die Trace-Datei von Protokoll-Monitor erfordern.

```
001550 *** FindFirstFileW 'C:\test\cmd_test\bin\foobar.*' -> INVALID_HANDLE_VALUE *** failed
[system probe
```

So führen Sie eine erweiterte Untersuchung des cmd.exe-Eintrags aus

1. Um zu ermitteln, warum das Dienstprogramm `cmd.exe c:\test\cmd_test\bin` sondiert, überprüfen Sie das Protokoll für diese Protokolleintragsnummer und stellen Sie fest, was sich vor dem Aufruf ereignete.
2. Um die Speicherorte zu ermitteln, wo das Dienstprogramm `cmd.exe` den Pfad `c:\test\cmd_test path` abruft, überprüfen Sie das Protokoll auf die Einträge `GetCurrentDirectoryW` und `GetFullPathNameW`.

```
000861 0a88 cmd.exe  :4ad01580->USERENV.dll :769c0396 GetCurrentDirectoryW (IN DWORD
nBufferLength=104h)
000862 0a88                     GetCurrentDirectoryW -> 0x14 (C:\test\cmd_test\bin)
000863 0a88 cmd.exe  :4ad01580<-USERENV.dll :769c0396 GetCurrentDirectoryW ->DWORD=14h
(OUT LPWSTR lpBuffer=*4AD34400h->L"C:\test\cmd_test\bin")
000864 0a88 cmd.exe  :4ad05b74->ole32.dll :774e03f0 Getfile type (IN HANDLE hFile=7h)
000865 0a88                     Getfile type 7 -> 0x2
000866 0a88 cmd.exe  :4ad05b74<-ole32.dll :774e03f0 Getfile type ->DWORD=2h ()
.
.
001533 0a88 cmd.exe  :4ad01b0d<-kernel32.dll:7c80ac0f SetErrorMode ->UINT=0h ()
001534 0a88 cmd.exe  :4ad01b13->kernel32.dll:7c80ac0f SetErrorMode (IN UINT uMode=1h)
001535 0a88 cmd.exe   :4ad01b13<-kernel32.dll:7c80ac0f SetErrorMode ->UINT=0h ()
001536 0a88 cmd.exe  :4ad01b24->IMM32.DLL :7639039b GetFullPathNameW (IN LPCWSTR
lpFileName=*1638C0h->L"," IN DWORD nBufferLength=208h)
001537 0a88                     GetFullPathNameW . -> 20 (buf=C:\test\cmd_test\bin,
file_part=bin)
001538 0a88 cmd.exe  :4ad01b24<-IMM32.DLL :7639039b GetFullPathNameW ->DWORD=14h
(OUT LPWSTR lpBuffer=*163D60h->L"C:\test\cmd_test\bin," OUT *lpFilePart=*13D8D4h-
>*163D82h->L"bin")
.
.
```

```
001549 0a88 cmd.exe   :4ad01b5f->USERENV.dll :769c03fa FindFirstFileW (IN LPCWSTR
lpFileName=*1638C0h->L"C:\test\cmd_test\bin\foobar.*")
001550 0a88                    FindFirstFileW 'C:\test\cmd_test\bin\foobar.*' ->
INVALID_HANDLE_VALUE *** failed [system probe C:\test\cmd_test\bin\foobar.* -> ffffffffh][no
virtual or system matches]
```

Das Dienstprogramm `cmd.exe` erhält den ersten Speicherort durch den Aufruf `GetCurrentDirectoryW` und den zweiten Speicherort durch den Aufruf `GetFullPathNameW`, wobei "." den Pfad festlegt. Diese Aufrufe liefern den Pfad für das aktuelle Arbeitsverzeichnis zurück. Die Protokolldatei zeigt, dass das Dienstprogramm `cmd.exe` die Aufforderung `c:\test\cmd_test\bin>` erstellt. Das Dienstprogramm fragt die `PROMPT`-Umgebungsvariable ab, die `$P$G` zurückliefert, und verwendet die API-Funktion `WriteConsoleW`, um die Eingabeaufforderung auf dem Bildschirm anzuzeigen, nachdem `$P$G` intern zu `c:\test\cmd_test\bin>` erweitert wurde.

Problembehandlung bei bestimmten Anwendungen

Tipps für die Problembehandlung stehen für die Kapselung von Microsoft Outlook, Explorer.exe und Java Runtime Environment zur Verfügung.

Fehlerbehebung beim Setup der Registrierung für Microsoft Outlook

Microsoft Outlook speichert Kontoeinstellungen in Registrierungsschlüsseln und -dateien. Wenn Sie Microsoft Outlook zum ersten Mal starten, prüft das Programm, ob die Schlüssel vorhanden sind. Kann Microsoft Outlook die Schlüssel nicht lokalisieren, werden Sie aufgefordert, ein neues Konto zu erstellen.

Dieser Vorgang funktioniert in der virtuellen Umgebung fehlerfrei, wenn Microsoft Outlook nicht auf dem physischen System installiert ist. Hat der Anwender Microsoft Outlook bereits auf dem physischen System installiert, findet die gekapselte Version die Registrierungsschlüssel in der Systemregistrierung und verwendet diese Einstellungen. Sie müssen den Isolationsmodus „Full (Voll)“ für die Registrierungsschlüssel und -dateien verwenden, in denen Outlook seine Einstellungen speichert.

So legen Sie den Isolationsmodus „Full (Voll)“ für die Registrierungsschlüssel von Microsoft Outlook fest

1 Fügen Sie zur Datei `HKEY_CURRENT_USER.txt` die folgenden Einträge hinzu:

```
isolation_full HKEY_CURRENT_USER\Identities
isolation_full HKEY_CURRENT_USER\Software\Microsoft\Windows
NT\CurrentVersion\Windows Messaging Subsystem\Profiles
```

2 Erstellen Sie eine `##Attributes.ini`-Datei mit den folgenden Einträgen:

```
[Isolation]
DirectoryIsolationMode=Full
```

3 Platzieren Sie die `##Attributes.ini`-Datei in jedem der folgenden Unterverzeichnisse:

```
%AppData%\Microsoft\AddIns
%AppData%\Microsoft\Office
%AppData%\Microsoft\Outlook
%Local AppData%\Microsoft\FORMS
%Local AppData%\Microsoft\Outlook
```

4 (Optional) Erstellen Sie die Unterverzeichnisse, wenn sie nicht vorhanden sind.

Anzeigen von Anhängen in Microsoft Outlook

Sobald Sie einen Anhang zum Betrachten öffnen, erstellt Microsoft Outlook ein Standardverzeichnis zum Speichern von Anhängen. Der übliche Speicherort lautet: `C:\Dokumente und Einstellungen\<Benutzer_Name>\Lokale Einstellungen\Temp\Temporary Internet Files\OLK<xxxx>`. Die letzten `xxxx` werden durch einen nach dem Zufallsprinzip erstellten Eintrag ersetzt.

Sie können Anhänge betrachten, wenn die Anwendung zum Betrachten in derselben virtuellen Sandbox wie Microsoft Outlook ausgeführt wird. Externe Anwendungen können möglicherweise die Datei zur Anzeige nicht finden, da Microsoft Outlook die Datei in der Sandbox speichert. Sie müssen den Isolationsmodus „Merged (Zusammengeführt)“ verwenden für das Verzeichnis, das die Anhänge speichert.

So legen Sie den Isolationsmodus „Merged (Zusammengeführt)“ für die Anzeige von Anhängen in Microsoft Outlook fest

1 Fügen Sie zur Datei `HKEY_CURRENT_USER.txt` einen Wert hinzu, der den Namen des Anhangverzeichnisses festlegt:

```
isolation_full
HKEY_CURRENT_USER\Software\Microsoft\Office\11.0\Outlook\Security
Value=OutlookSecureTempFolder
REG_SZ~%Profile%\Lokale Einstellungen\OutlookTempxxxx#2300
```

In diesem Beispiel steht 11.0 im Schlüsselnamen für Microsoft Outlook 2003.

2 Ersetzen Sie zur Erhöhung der Sicherheit die letzten vier `xxxx`-Zeichen durch zufällige alphanumerische Zeichen.

3 Erstellen Sie ein Verzeichnis, das im Registrierungsschlüssel `OutlookSecureTempFolder` Ihres ThinApp-Projekts benannt wird.

Erstellen Sie zum Beispiel das Verzeichnis `%Profile%\Lokale Einstellungen\OutlookTempxxxx`.

4 Erstellen Sie im Verzeichnis `%Profile%\Lokale Einstellungen\OutlookTempxxxx` eine `##Attributes.ini`-Datei mit den folgenden Einträgen:

```
[Isolation]
DirectoryIsolationMode=Merged
```

Starten von Explorer.exe in der virtuellen Umgebung

Durch das Ausführen einer Instanz des Dienstprogramms `explorer.exe` auf einem Windows-Betriebssystem wird das Hinzufügen eines Einstiegspunkts zu Windows Explorer und das Starten des Programms innerhalb der virtuellen Umgebung erschwert.

Sie können mithilfe der folgenden Methoden innerhalb der virtuellen Umgebung ein Fenster von Windows Explorer öffnen:

- Fügen Sie zu iExplorer einen Einstiegspunkt hinzu und starten Sie die Datei mit dem –E-Parameter.

  Fügen Sie beispielsweise der Datei `Package.ini` die folgenden Einträge hinzu:

  ```
  [iexplore.exe]
  Shortcut=xxxx.exe
  Source=%ProgramFilesDir%\Internet Explorer\iexplore.exe
  CommandLine=%ProgramFilesDir%\Internet Explorer\iexplore.exe –E
  ```

- Fügen Sie den folgenden virtuellen Registrierungsschlüssel hinzu:

  ```
  isolation_full HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer
  Value=DesktopProcess
  REG_DWORD=#01#00#00#00
  ```

- Fügen Sie zur Datei `Package.ini` die folgenden Einträge hinzu:

  ```
  [explorer.exe]
  Shortcut=xxxxxx.exe
  Source=%SystemROOT%\explorer.exe
  ```

  Verwenden Sie diese Methode, um das virtuelle Dateisystem mit einer bekannten Oberfläche zu durchsuchen, und aktivieren Sie Dateitypzuordnungen ohne Systemänderungen, insbesondere bei der Verwendung von portablen Anwendungen. Sie können ohne Systemänderungen auf die Shell-integrierten Komponenten zugreifen.

Problembehandlung bei Versionskonflikten von Java Runtime Environment

Ein Konflikt kann auftreten, wenn eine Java-Version auf dem physischen System installiert ist und eine andere Version in einer gekapselten ausführbaren Datei enthalten ist. Aktualisierte Versionen von Java installieren eine Plug-In-DLL, die Internet Explorer lädt. Diese Plug-In-DLL überschreibt virtuelle Registrierungsschlüssel und führt zu Konflikten mit virtuellen Kopien älterer Java-Laufzeiten.

So verhindern Sie das Laden von Plug-In-DLLs durch Internet Explorer

Fügen Sie am Anfang der Datei `HKEY_LOCAL_MACHINE.txt` folgenden Eintrag hinzu:

```
isolation_full HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\Browser
Helper Objects
```

Konfigurieren von Paketparametern

A

Fortgeschrittene Anwender können die Parameter der virtuellen Anwendung außerhalb des Kapselungsprozesses anpassen. Die `Package.ini`-Datei befindet sich im Projektordner und enthält Parameter zur Konfiguration einer gekapselte Anwendung während des Build-Prozesses. Damit die Parameteränderungen wirksam werden, müssen Sie die `Package.ini`-Datei speichern und das Projekt erstellen.

Parameter können sich auf die Konfiguration von Isolationsmodi und Build-Optionen auswirken, darunter die Einstellungen für MSI, Application Link, Application Sync und für Einstiegspunkte von Anwendungen. Der Setup Capture-Assistent legt die anfänglichen Werte bestimmter `Package.ini`-Parameter fest. Siehe „Kapseln einer Anwendung mit dem Setup Capture-Assistenten“ auf Seite 16.

Dieser Abschnitt umfasst die folgenden Themen:

- „Package.ini-Dateistruktur“ auf Seite 66
- „Parameter, die auf Package.ini- oder ##Attributes.ini-Dateien angewendet werden“ auf Seite 66
- „Konfigurieren der ThinApp-Laufzeit“ auf Seite 66
- „Konfigurieren des Dateisystems und der Registrierungsisolation“ auf Seite 68
- „Konfigurieren von Datei- und Protokollzuordnungen“ auf Seite 69
- „Konfigurieren der Build-Ausgabe“ auf Seite 70
- „Konfigurieren von Berechtigungen und Sicherheit“ auf Seite 71
- „Konfigurieren von Objekten und DLL-Dateien“ auf Seite 73
- „Konfigurieren von Speicherplatz“ auf Seite 77
- „Konfigurieren von Prozessen und Diensten“ auf Seite 79
- „Konfigurieren von Datei- und Blockgrößen“ auf Seite 81
- „Konfigurieren von Symbolen“ auf Seite 82
- „Konfigurieren der Protokollierung“ auf Seite 83
- „Konfigurieren von Versionen“ auf Seite 84
- „Konfigurieren von lokalen Informationen“ auf Seite 85
- „Konfigurieren von einzelnen Anwendungen“ auf Seite 86
- „Konfigurieren von abhängigen Anwendungen mit Application Link“ auf Seite 89
- „Konfigurieren von Anwendungs-Updates mit Application Sync“ auf Seite 92
- „Konfigurieren von MSI-Dateien“ auf Seite 94
- „Konfigurieren von Sandbox-Speicher und Bestandsnamen“ auf Seite 98

Package.ini-Dateistruktur

Die Sektion `[BuildOptions]` der `Package.ini`-Datei gilt für alle Anwendungen. Die einzelnen Anwendungen übernehmen diese Parameter, sofern keine anwendungsspezifischen Einträge Vorrang vor diesen Einstellungen haben. Zum Beispiel kann der Abschnitt `[Adobe Reader 8.exe]` in der `Package.ini`-Datei für eine Adobe Reader-Anwendung Einstellungen enthalten, die Vorrang gegenüber den allgemeineren `[BuildOptions]`-Parametern haben. Die anwendungsspezifischen Parameter weisen die Einstiegspunkte für die Anwendungen aus, die Sie während des Build-Prozesses erstellen.

Package.ini-Parameterplatzierung

Die in eckigen Klammern gesetzten Sektionen der `Package.ini`-Datei erfordern, dass die Parameter unter der geeigneten Sektion stehen. Diese Sektionen tragen folgende Überschriften:

- `[BuildOptions]`
- `[<Anwendung>.exe]`
- `[FileList]`
- `[Compression]`
- `[Isolation]`

Parameter, die nicht zu diesen Sektionen passen, müssen nicht unter eine bestimmte Überschrift geordnet sein. Parameter müssen nicht in alphabetischer Reihenfolge stehen. Die Parameter `[FileList]`, `[Compression]` und `[Isolation]` fungieren als `[BuildOptions]`-Parameter, sind jedoch aus Gründen der Abwärtskompatibilität gesondert gruppiert. Sie können die Überschrift `[FileList]` manuell zur Datei hinzufügen, wenn Sie den Parameter `ExcludePattern` hinzufügen müssen.

Parameter, die auf Package.ini- oder ##Attributes.ini-Dateien angewendet werden

Sie können die Parameter `DirectoryIsolationMode`, `Compression` und `ExcludePattern` in einer `##Attributes.ini`-Datei verwenden, um die `Package.ini`-Einstellungen auf der Verzeichnisebene zu überschreiben. Die `##Attributes.ini`-Datei ist in den Ordnermakros des Projektordners vorhanden. Weitere Informationen über die `##Attributes.ini`-Datei erhalten Sie unter „Ändern der Einstellungen in der ##Attributes.ini-Datei“ auf Seite 24.

Konfigurieren der ThinApp-Laufzeit

Die Aufgaben der Laufzeitkonfiguration schließen Parameter ein, die sich auf die Leistung beim Anwendungsstart und auf virtuelle Computernamen beziehen.

NetRelaunch

Der Parameter `NetRelaunch` legt fest, ob eine Anwendung von der lokalen Festplatte aus neu gestartet werden soll, wenn die Anwendung von einer Netzwerkfreigabe oder von einem Wechseldatenträger, wie einem USB-Laufwerk, aus ausgeführt wird. Dieser Parameter ist nützlich, wenn Sie den langsamen Start oder die Leistung von Anwendungen korrigieren müssen.

ThinApp erkennt, ob eine Anwendung von einem Netzlaufwerk oder von einem Wechseldatenträger aus ausgeführt wird und verwendet eine ausführbare Stub-Datei, um einen Neustart der Anwendung von der lokalen Festplatte aus durchzuführen. Dieser Prozess löst Leistungsprobleme, die entstehen, wenn Symantec AntiVirus versucht, eine vollständige Überprüfung für ausführbare Dateien, die von einer Netzwerkfreigabe oder von einem Wechseldatenträger gestartet werden, sowie für ausführbare Dateien, die die anfänglichen Netzwerkverbindungen herstellen, durchzuführen. Die Überprüfung kann sich auf die Startdauer von großen ausführbaren Dateien auswirken.

Da zahlreiche Desktop-Computer mit Symantec AntiVirus ausgestattet sind, ermöglicht ThinApp, dass die Anwendungen von einer Netzwerkfreigabe aus gestartet werden, ohne dass Verzögerungen durch die Überprüfung entstehen. Wird die Anwendung von einer Netzwerkfreigabe oder von einem Wechseldatenträger aus ausgeführt, erstellt ThinApp eine ausführbare Stub-Datei in dem Verzeichnis, das der Parameter `CachePath` auf der lokalen Festplatte festlegt und startet die Anwendung von dieser ausführbaren Stub-Datei aus neu. Die ausführbare Stub-Datei kann die Laufzeit vom großen Paket laden und den Rest der Anwendung von ihrem ursprünglichen Netzwerkspeicherort lesen. Symantec AntiVirus sieht die Anwendung als lokal an, und überprüft die größere ausführbare Datei der Netzwerkfreigabe oder auf dem Wechseldatenträger nicht.

Beispiele

Wird die Anwendung von einem Netzlaufwerk oder von einem Wechseldatenträger aus gestartet, erstellt der Standardwert des Parameters `NetRelaunch` eine lokale Stub-Datei, um einen Neustart der Anwendung durchzuführen.

```
[BuildOptions]
NetRelaunch=1
```

Wenn Ihre Anwendung klein ist oder wenn Sie sicher sind, dass Symantec AntiVirus nicht auf den Desktop-Computern installiert ist, auf denen die Anwendung ausgeführt werden soll, können Sie den Parameter `NetRelaunch` deaktivieren, um eine bessere Leistung beim anfänglichen Starten zu erzielen.

```
[BuildOptions]
NetRelaunch=0
```

RuntimeEULA

Der Parameter `RuntimeEULA` steuert die Anzeige der Endbenutzerlizenzvereinbarung (EULA) für das Paket. Dieser Parameter bezieht sich auf ältere EULA-Anforderungen.

VMware erfordert keine Laufzeit-EULA für ThinApp-Pakete. Ändern Sie den Wert dieses Parameters nicht.

Beispiele

Der Standardwert für den Parameter `RuntimeEULA` verhindert die Anzeige der Endbenutzerlizenzvereinbarung.

```
[BuildOptions]
;Default: do not show an Eula (;Standard: Endbenutzerlizenzvereinbarung nicht anzeigen)
RuntimeEULA=0
```

Sie können eine Endbenutzerlizenzvereinbarung anzeigen.

```
[BuildOptions]
;Turn on display of EULA (;Anzeige der Endbenutzerlizenzvereinbarung aktivieren)
RuntimeEULA=1
```

VirtualComputerName

Der Parameter `VirtualComputerName` virtualisiert den Computernamen. Dies ist nützlich für eine Bereitstellungsmaschine, die nicht denselben Namen hat wie der Kapselungsmaschine.

Anwendungen können den Namen des Rechners, auf dem sie installiert wurden oder über den sie auf eine Datenbank zugreifen, und den Namen des Computers in der Verbindungszeichenfolge verwenden. Bei gekapselten Anwendungen ist der Computername virtuell. Hierdurch soll sichergestellt werden, dass die Anwendung auf allen Rechnern ausgeführt werden kann.

Dieser Parameter ist eine Zeichenfolge, die von den API-Funktionen `GetComputerName` und `GetComputerNameEx` in einer gekapselten Anwendung ausgegeben wird.

Beispiele

Trägt der Computer, mit dem die Kapselung ausgeführt wurde, nicht den Namen LOCALHOST, kommentiert ThinApp den Parameter aus.

```
;VirtualComputerName=<Originaler_Maschinenname>
```

Wenn Sie eine neu aufgesetzte Maschine in `LOCALHOST` umbenennen, bevor Sie den Kapselungsprozess ausführen, aktiviert die `Package.ini`-Datei den Eintrag `VirtualComputerName`.

```
VirtualComputerName=LOCALHOST
```

Wenn Sie den Befehl `GetComputerName` oder `GetComputerNameEx` eingeben, gibt der Rechner `LOCALHOST` aus. Wenn das Windows-System die API-Funktionen `GetComputerName` und `GetComputerNameEx` benötigt, um standardmäßig zu funktionieren und den tatsächlichen Namen auszugeben, auf dem die Anwendung ausgeführt wird, sollte der Rechner nicht in `LOCALHOST` umbenannt werden.

Abgesehen von der Angabe einer literalen Zeichenfolge wie LOCALHOST können Sie auch eine Umgebungsvariable angeben.

```
VirtualComputerName=%VCOMPNAME%
```

Wenn Sie eine Umgebungsvariable angeben, wird als Wert der Wert der Umgebungsvariablen ausgegeben. Lautet der Wert des Parameters `VirtualComputerName` %VCOMPNAME%, und ist als Umgebungsvariable für %VCOMPNAME% der Wert `EnvCompName` angegeben, so gibt die API-Funktion `GetComputerName` den Wert `EnvCompName` aus.

Wow64

Der Parameter `Wow64` simuliert eine 32-Bit-Umgebung für 32-Bit-Anwendungen, die nicht auf einem 64-Bit-Windows-Betriebssystem ausgeführt werden können. Versucht eine 32-Bit-Anwendung, eine eigene 64-Bit-Registrierungsumleitung zu verarbeiten, so können Sie diesen Parameter vor der Erstellung eines Projekts aktivieren.

Beispiele

Sie können den Parameter auskommentiert lassen, um die Emulation von Windows auf Windows 64-Bit (WOW64) zu verhindern.

```
[BuildOptions]
;Wow64=0
```

Sie können auf einem 64-Bit-Betriebssystem für 32-Bit-Anwendungen eine 32-Bit-Umgebung simulieren. Beispielsweise funktioniert eine virtualisierte 32-Bit-Oracle-Anwendung möglicherweise nicht auf einem 64-Bit-Betriebssystem.

```
[BuildOptions]
Wow64=0
```

Konfigurieren des Dateisystems und der Registrierungsisolation

Isolationsmodusparameter bestimmen den Schreibzugriff auf das Dateisystem und die Registrierungsschlüssel.

DirectoryIsolationMode

Der Parameter `DirectoryIsolationMode` gibt den Grad des Schreibzugriffs für Verzeichnisse unter dem physischen Dateisystem an. ThinApp bietet die Isolationsmodi: WriteCopy, Merged (Zusammengeführt) und Full (Voll). Dieser Parameter steuert den Standardisolationsmodus für die Verzeichnisse des Anwendungspakets, für die keine spezifischen Einstellungen vorliegen. Die Standardeinstellung richtet sich jeweils nach der Kapselung der Anwendung.

Die `Package.ini`-Datei legt den Standardisolationsmodus für das betreffende Projekt fest. Einzelne `##Attributes.ini`-Dateien haben Vorrang gegenüber der `Package.ini`-Datei und geben den Isolationsmodus für bestimmte Verzeichnisse und untergeordnete Verzeichnisse an. Alle nicht angegebenen Verzeichnisse, wie zum Beispiel `C:\myfolder`, übernehmen den Isolationsmodus von der `Package.ini`-Datei. Sie müssen vor diesem Parametereintrag die Überschrift `[Isolation]` hinzufügen.

Verwenden Sie den Isolationsmodus Full (Voll) nicht in der `Package.ini`-Datei, da der Modus die Fähigkeit System-DLLs zu erkennen und zu laden, blockiert. Sie können den Isolationsmodus Full (Voll) als Überschreibmechanismus in den `##Attributes.ini`-Dateien verwenden.

Informationen zu den Definitionen und den Effekt der Isolationsmodi erhalten Sie unter „Festlegen des Dateisystemzugriffs“ auf Seite 19.

Beispiele

Mithilfe des WriteCopy-Isolationsmodus kann die Anwendung Ressourcen auf dem lokalen Rechner lesen; sie kann jedoch nicht auf den Hostcomputer schreiben. Dies ist die Standardeinstellung für das Dienstprogramm `snapshot.exe`.

```
[Isolation]
DirectoryIsolationMode=WriteCopy
```

Mithilfe des Isolationsmodus „Merged (Zusammengeführt)“ kann die Anwendung Ressourcen an einem beliebigen Speicherort auf dem Computer lesen und an einen beliebigen Speicherort auf dem Computer schreiben, außer wo das Paket etwas Gegenteiliges angibt. Dies ist die Standardeinstellung für den Setup Capture-Assistenten.

```
[Isolation]
DirectoryIsolationMode=Merged
```

RegistryIsolationMode

Der Parameter `RegistryIsolationMode` steuert den Standardisolationsmodus für Registrierungsschlüssel im Paket. Diese Einstellung gilt für die Registrierungsschlüssel, die keine expliziten Einstellungen aufweisen.

Weitere Informationen zu den Definitionen der Isolationsmodi erhalten Sie unter „Festlegen des Dateisystemzugriffs“ auf Seite 19. Registrierungsisolationsmodi sind nur in der `Package.ini`-Datei vorhanden. Diese Einstellung kann mithilfe des Setup Capture-Assistenten nicht konfiguriert werden.

Beispiele

Mit dem Parameter `RegistryIsolationMode` können Sie sicherstellen, dass die Anwendung Schlüssel vom Hostcomputer lesen, jedoch nicht auf den Hostcomputer schreiben kann. Wenn Sie den Registrierungs-Isolationsmodus in der `Package.ini`-Datei nicht angeben, lautet der Standardwert `WriteCopy`.

```
[Isolation]
RegistryIsolationMode=WriteCopy
```

Mit dem Parameter `RegistryIsolationMode` können Sie sicherstellen, dass die Anwendung in alle Schlüssel auf dem Computer schreiben kann, außer diejenigen, für die das Paket etwas Gegenteiliges angibt.

```
[Isolation]
RegistryIsolationMode=Merged
```

Konfigurieren von Datei- und Protokollzuordnungen

Datei- und Protokollparameter ordnen Anwendungen Dateierweiterungen zu und legen Protokolle fest, die für die physische Umgebung sichtbar sind.

FileTypes

Der Parameter `FileTypes` listet Dateierweiterungen auf, die vom Dienstprogramm `thinreg.exe` mit einer ausführbaren Datei verknüpft werden. Beim Kapselungsprozess werden Standardwerte generiert, denen Sie keine weiteren hinzufügen können. Sie können Dateierweiterungen entfernen, die nicht mit dem virtuellen Paket verknüpft werden sollen. Geben Sie keine Trennzeichen zwischen den Dateierweiterungen in der Liste an.

Beispiele

Der Standardwert für ein Paket von Microsoft Word 2007 ist `.doc.docx`. Wenn Sie Microsoft Office 2007 virtualisieren und Microsoft 2003 in der physischen Umgebung installiert ist, können Sie die Dateierweiterung `.doc` aus der `FileTypes`-Liste entfernen und die Dateierweiterung `.docx` beibehalten, um sicherzustellen, dass `.doc`-Dateien mit Microsoft Word 2003 geöffnet werden und dass `.docx`-Dateien mit Microsoft Word 2007 geöffnet werden.

```
[Microsoft Office Word 2007.exe]
FileTypes=.docx
```

Der Kapselungsprozess kann Dateitypzuordnungen für die Dateierweiterungen `.doc` und `.dot` erstellen und diese mit Microsoft Word verknüpfen.

```
[Microsoft Office Word 2003.exe]
ReadOnlyData=bin\Package.ro.tvr
Source=%ProgramFilesDir%\Microsoft Office\OFFICE11\WINWORD.EXE
FileTypes=.doc.dot
```

Protocols

Der Parameter `Protocols (Protokolle)` gibt die Protokolle an, die für Anwendungen in der physischen Umgebung sichtbar sind, zum Beispiel HTTP. Dieser Parameter ist dem Parameter `FileTypes` ähnlich, bezieht sich aber auf Anwendungen, die Protokolle statt Dateitypen behandeln. Beim Kapselungsprozess werden Standardwerte generiert, denen Sie keine weiteren hinzufügen können. Für Browser oder andere Anwendungen können Sie Einträge entfernen.

Beispiele

Mit dem Kapselungsprozess können Protokolle, wie das Protokoll `mailto` für ein Microsoft Outlook-Paket angegeben werden.

```
[Microsoft Office Outlook 2007.exe]
Protocols=feed;feeds;mailto;Outlook.URL.mailto;stssync;webcal;webcals
```

Konfigurieren der Build-Ausgabe

Build-Parameter geben den Speicherort der Build-Ausgabe und die Dateien an, die aus dem Paket ausgeschlossen werden.

OutDir

Der Parameter `OutDir` gibt das Verzeichnis an, in dem die `build.bat`-Ausgabe gespeichert wird. Ändern Sie den Wert dieses Parameters nicht.

Beispiele

Der Standard- und erforderliche Wert gibt das `bin`-Verzeichnis des Projekts an.

```
[BuildOptions]
OutDir=bin
```

ExcludePattern

Der Parameter `ExcludePattern` schließt Dateien bzw. Verzeichnisse während des Build-Prozesses für die Anwendung aus. Sie müssen vor diesem Parametereintrag die Überschrift `[FileList]` hinzufügen.

Sie können ein Komma verwenden, um die Muster in der Liste zu trennen. Platzhalter (*)können mit keinem der Zeichen oder mindestens mit einem der Zeichen übereinstimmen, und Fragezeichen (?) stimmen mit genau einem Zeichen überein. Die Syntax ist vergleichbar mit dem `dir`-Befehl in DOS; jedoch können Sie Platzhalterzeichen auf Verzeichnis- und Dateinamen anwenden.

Sie können den Parameter `ExcludePattern` in der `Package.ini`-Datei angeben. Der Musterausschluss wird dann auf die gesamte Verzeichnisstruktur angewandt. Sie können den Parameter auch in der `##Attributes.ini`-Datei angeben. In diesem Fall fügt ThinApp den Musterausschluss in die aktuelle Liste mit Ausschlüssen wendet, aber die Einstellungen nur auf das spezifische Verzeichnis und dessen Unterverzeichnisse an. Sie können unterschiedliche Ausschlusslisten für verschiedene Verzeichnisse in Ihrem Projekt erstellen.

Beispiele

Wenn Sie Pakete in einem Versionskontrollsystem speichern und Sie die Versionskontrollinformationen aus dem virtuellen Dateisystem ausschließen möchten, können Sie alle `.svn` oder `.cvs` genannten Verzeichnisse und alle Unterverzeichnisse ausschließen.

```
[FileList]
ExcludePattern=\.svn,\.cvs
```

Das Muster stimmt nicht mit Dateinamen oder Verzeichnissen überein, die `.svn` oder `.cvs` in der Mitte der Zeichenfolge enthalten.

Sie können alle Pfade ausschließen, die auf `.bak` oder `.msi` enden.

```
[FileList]
ExcludePattern=*.bak,*.msi
```

Konfigurieren von Berechtigungen und Sicherheit

Sicherheitsaufgaben schließen Parameter ein, die den Benutzerzugriff auf Pakete definieren und den DEP-Schutz (Data Execution Prevention) ändern.

AccessDeniedMsg

Der Parameter `AccessDeniedMsg` enthält eine Fehlermeldung, die für Benutzer angezeigt wird, die zur Ausführung eines Pakets nicht berechtigt sind. Die Standardeinstellung lautet `You are not currently authorized to run this application. Please contact your Administrator. (Sie sind zur Ausführung dieser Anwendung zurzeit nicht berechtigt. Bitte wenden Sie sich an den Administrator.)`

Beispiele

Die Zeichenfolge `AccessDeniedMsg` kann mit einer Nummer für den technischen Support ergänzt werden.

```
[BuildOptions]
PermittedGroups=Administrator;Office-Benutzer
AccessDeniedMsg=Sie sind zur Ausführung dieser Anwendung zurzeit nicht berechtigt. Bitte wenden
Sie sich an den Support unter der Rufnummer 1-800-822-2992
```

AddPageExecutePermission

Der Parameter `AddPageExecutePermission` bezieht sich auf Anwendungen, die in einer DEP-Umgebung (Data Execution Prevention) nicht funktionieren.

Die DEP-Funktion von Windows XP SP2, Windows Server 2003 und höheren Versionen des Betriebssystems schützt vor bestimmten Sicherheitslücken, die bei Pufferüberlauf auftreten. Diese Funktion schafft bestimmte Kompatibilitätsprobleme. Windows schaltet die Funktion standardmäßig bei Windows XP SP2 aus und Sie können eine rechnerspezifische Auswahlliste mit den Anwendungen verwenden, auf die der DEP-Schutz angewandt werden soll. Die Auswahlregeln können schwierig zu verwalten sein, wenn eine große Anzahl an Rechner und Anwendungen beteiligt sind. Der Parameter `AddPageExecutePermission` weist ThinApp an, eine Ausführungsberechtigung zu Seiten hinzuzufügen, die von einer Anwendung zugewiesen werden. Die Anwendung kann ohne vorherige Änderung der Auswahlliste auf Rechnern ausgeführt werden, bei denen der DEP-Schutz aktiviert ist.

Beispiele

Der Standardwert des Parameters `AddPageExecutePermission` verhindert jede Änderung des DEP-Schutzes.

```
[BuildOptions]
AddPageExecutionPermission=0
```

Sie können die Ausführungsberechtigung zu Seiten hinzufügen, die von einer Anwendung zugewiesen werden. ThinApp führt einen Code von Speicherseiten aus, die die Anwendung angibt. Dies ist nützlich für Anwendungen, die das Programm und seine Daten in einem Bereich des Arbeitsspeichers kombinieren.

```
[BuildOptions]
;Disable some Data Execution protections for this particular application (;Deaktivierung einiger
               DEP–Schutzmaßnahmen für diese spezielle Anwendung)
AddPageExecutionPermission=1
```

PermittedGroups

Der Parameter `PermittedGroups` beschränkt ein Paket auf eine bestimmte Gruppe von Active Directory-Benutzern. Sie können diesen Parameter unter der Überschrift `[BuildOptions]` verwenden, damit er sich auf das Paket auswirkt, oder unter der Überschrift `[<Anwendung>.exe]`, damit er sich auf eine bestimmte Anwendung auswirkt. Der Wert `[<Anwendung>.exe]` überschreibt den Standardwert `[BuildOptions]` dieser bestimmten Anwendung.

Sie können Gruppennamen, SID-Zeichenfolgen oder eine Mischung aus Gruppennamen und SID-Zeichenfolgen in derselben Zeile wie dem Parameter `PermittedGroups` angeben. Wenn Sie einen domänenbasierten Gruppennamen verwenden, müssen Sie während der Erstellung des Anwendungspakets mit der betreffenden Domäne verbunden sein. Wenn Sie eine SID direkt in den Parameterwert eingeben, brauchen Sie keine Verbindung zu der Domäne herzustellen, in der die SID definiert ist.

Der Parameter unterstützt keine verschachtelten Active Directory-Gruppen. Ist ein Benutzer beispielsweise Mitglied der Gruppe A und ist Gruppe A Mitglied der Gruppe B, so kann ThinApp den Benutzer nur als Mitglied der Gruppe A erkennen, jedoch nicht als Mitglied von Gruppe A und Gruppe B.

Wenn ThinApp eine Anwendung erstellt, geht ThinApp davon aus, dass alle angegebenen Gruppennamen gültig sind, und konvertiert die Namen in SID-Werte. ThinApp kann die Gruppenzugehörigkeit bei Laufzeit mithilfe von im Cache gespeicherten Benutzerdaten auflösen. Sie können Laptop-Benutzer weiterhin authentifizieren, selbst wenn diese nicht mit dem Netzwerk verbunden sind.

Hat der Benutzer keine Berechtigung zur Ausführung des Pakets, so können Sie den Parameter `AccessDeniedMsg` derart anpassen, dass der Benutzer einen entsprechenden Hinweis erhält.

Beispiele

Sie können eine Liste mit durch Semikolon getrennten Active Directory-Benutzergruppennamen angeben. Die Parameter [BuildOptions] legen die globalen Einstellungen für das gesamte Projekt fest.

```
[BuildOptions]
PermittedGroups=Administrator;Office–Benutzer
AccessDeniedMsg=Sie sind zur Ausführung dieser Anwendung zurzeit nicht berechtigt. Bitte wenden
Sie sich an den Support unter der Rufnummer 1–800–822–2992
```

Sie können eine Benutzergruppeneinstellung für eine bestimmte Anwendung angeben, die die globale Einstellung `PermittedGroups` überschreibt.

```
[App1.exe]
PermittedGroups=Gast
AccessDeniedMsg=Sie sind zur Ausführung dieser Anwendung zurzeit nicht berechtigt. Bitte wenden
Sie sich an den Support unter der Rufnummer 1–800–822–2992
```

Wenn Sie keine `PermittedGroups`-Einstellung für eine Anwendung angeben, übernimmt die Anwendung den globalen Wert `PermittedGroups` in der Sektion `[BuildOptions]`.

```
[App2.exe]
...
```

Sie können in einem Eintrag für den `PermittedGroups`-Parameter sowohl Gruppennamen als auch SID-Zeichenfolgen verwenden.

```
PermittedGroups=S-1-5-32-544;Office Users
```

UACRequestedPrivilegesLevel

Der Parameter `UACRequestedPrivilegesLevel` gibt Berechtigungen für Programme an, für die UAC-Informationen erforderlich sind. Dieser Parameter wirkt sich auf Benutzer aus, die mit Windows Vista oder höheren Betriebssystemversionen arbeiten.

Sie können die folgenden Werte verwenden:

- asInvoker

  Dieser Wert verwendet das Profil in Vista.

- requireAdministrator
- highestAvailable

  Dieser Wert verwendet die höchste verfügbare Berechtigung, mit der die UAC-Eingabeaufforderung verhindert werden kann.

Wenn Sie keine Berechtigungen angeben, ordnet ThinApp keinen Standardwert zu, sondern arbeitet gemäß der asInvoker-Einstellung.

Beispiele

Sie können angeben, dass für ein Programm Administratorrechte erforderlich sind.

```
[BuildOptions]
UACRequestedPrivilegesLevel=requireAdministrator
```

UACRequestedPrivilegesUIAccess

Windows Vista oder höhere Betriebssystemversionen schützen einige Elemente der Benutzeroberfläche. Unter normalen Umständen ist für virtuelle Anwendungen kein Zugriff auf geschützte Elemente erforderlich. Sie können den Wert `true` oder `false` zum Parameter `UACRequestedPrivilegesUIAccess` hinzufügen, um den Zugriff auf die Benutzeroberfläche anzugeben.

Der Parameter `UACRequestedPrivilegesUIAccess` ist vorhanden, um parallele Funktionen zum Microsoft-Anwendungsmanifest zu bieten, anstatt den Wert zu verändern.

Beispiele

Der Standardwert `false` stellt sicher, dass die virtuelle Anwendung nicht auf geschützte Elemente zugreifen kann.

```
[BuildOptions]
UACRequestedPrivilegesUiAccess=false
```

Konfigurieren von Objekten und DLL-Dateien

ThinApp-Parameter können verwendet werden, um den COM-Objektzugriff und die Anforderungen zum Laden von DLLs anzugeben.

ExternalCOMObjects

Der Parameter `ExternalCOMObjects` steuert, ob ThinApp oder Windows einen CLSID-Schlüssel für ein spezifisches COM-Objekt erstellt.

Standardmäßig erstellt ThinApp alle COM-Objekte in der virtuellen Umgebung. COM unterstützt ausführbare Server außerhalb von Prozessen und dienstbasierte COM-Objekte. Wenn eine Anwendung COM-Objekte erstellen kann, die Modifizierungen auf dem Hostcomputer generieren, ist die Integrität des Hostcomputers in Gefahr. Wenn ThinApp außerhalb der Prozesse und dienstbasierten COM-Objekte in der virtuellen Umgebung ausgeführt wird, speichert ThinApp alle von den COM-Objekten durchgeführten Änderungen in der Sandbox.

Beispiele

Sie können ThinApp anweisen, zwei COM-Objekte außerhalb der virtuellen Umgebung auszuführen, wenn die Anwendung die Objekte erstellt.

```
[BuildOptions]
ExternalCOMObjects={8BC3F05E-D86B-11D0-A075-00C04FB68820};{7D096C5F-AC08-4F1F-BEB7-5C22C517CE39}
```

ExternalDLLs

Der Parameter `ExternalDLLs` kann erzwingen, dass Windows DLL-Dateien aus dem virtuellen Dateisystem lädt.

ThinApp lädt DDL-Dateien aus dem virtuellen Dateisystem und übergibt den Ladeprozess für DLL-Dateien auf dem physischen Dateisystem an Windows. Unter bestimmten Umständen muss Windows eine DLL-Datei im virtuellen Dateisystem laden. Zum Beispiel kann es vorkommen, dass sich eine DLL-Datei mithilfe von Windows-Hooks in andere Prozesse einfügt. Die DLL-Datei, die den Haken implementiert, muss auf dem Host-Dateisystem verfügbar sein und Windows muss die Datei laden. Wenn Sie eine DLL-Datei im Parameter `ExternalDLLs` angeben, extrahiert ThinApp die Datei aus dem virtuellen Dateisystem in die Sandbox und weist Windows an, die Datei zu laden.

Der Parameter `ExternalDLLs` unterstützt keine DLL-Dateien, die innerhalb des virtuellen Dateisystems von anderen DLL-Dateien abhängig sind. In diesem Fall kann Windows die DLL-Datei nicht laden.

Beispiele

Sie können ThinApp anweisen, den Ladeprozess der Dateien `inject.dll` und `injectme2.dll` an Windows zu übergeben.

```
[BuildOptions]
ExternalDLLs=inject.dll;injectme2.dll
```

IsolatedMemoryObjects

Der Parameter `IsolatedMemoryObjects` listet die gemeinsam genutzten Arbeitsspeicherobjekte auf, die von anderen Anwendungen isoliert werden sollen.

Anwendungen, die die Windows-Funktionen `CreateFileMapping` und `OpenFileMapping` verwenden, erstellen gemeinsam genutzte Arbeitsspeicherobjekte. Gemeinsam genutzte Arbeitsspeicherobjekte können benannt werden oder anonym bleiben. Benannte Objekte sind für andere Anwendungen sichtbar, die auf demselben Benutzerkonto ausgeführt werden. Möglicherweise möchten Sie gemeinsam genutzte Arbeitsspeicherobjekte isolieren, um sicherzustellen, dass virtuelle Anwendungen und Systemobjekte sich gegenseitig nicht erkennen können.

ThinApp isoliert gemeinsam genutzte Arbeitsspeicherobjekte, die von eingebetteten Internet Explorer-Instanzen verwendet werden. Zwischen den Dienstprogrammen `explorer.exe` und `iexplore.exe` tritt ein Konflikt auf, wenn die Dienstprogramme Sandbox-Dateien zuordnen. Mit dem Parameter `IsolatedMemoryObjects` können Sie weitere benannte gemeinsam genutzte Arbeitsspeicherobjekte isolieren, um sicherzustellen, dass die Objekte nur für andere virtuelle Anwendungen sichtbar sind, die dieselbe Sandbox verwenden.

Der Parameter `IsolatedMemoryObjects` akzeptiert eine Liste mit durch Semikolon (;) getrennten Einträgen. Jeder Eintrag kann die Platzhalter Sternchen (*) und Fragezeichen (?) enthalten, um einen Abgleich mit variablen Mustern durchzuführen.

Beispiele

Sie können zwei gemeinsam genutzte Arbeitsspeicherobjekte isolieren, ein Objekt mit `outlook` im Namen zuordnen und ein Objekt mit dem exakten Namen `My Shared Object (Mein gemeinsam genutztes Objekt)` zuordnen.

```
[BuildOptions]
IsolatedMemoryObjects=*outlook*;My Shared Object (Mein gemeinsam genutztes Objekt)
```

IsolatedSynchronizationObjects

Der Parameter `IsolatedSynchronizationObjects` listet spezifische Synchronisierungsobjekte auf, die von anderen Anwendungen isoliert werden sollen.

Windows enthält die folgenden benannten Synchronisierungsobjekte:

- `Mutex`

  Der Zugriff auf dieses Objekt erfolgt über OpenMutex und CreateMutex.

- `Semaphore`

  Der Zugriff auf dieses Objekt erfolgt über OpenSemaphore und CreateSemaphore.

- `Events`

  Der Zugriff auf dieses Objekt erfolgt über OpenEvent und CreateEvent.

Wenn eine Anwendung nicht funktioniert oder ein Fehler auftritt, müssen Sie diese Objekte möglicherweise in der virtuellen Umgebung isolieren, um eine Kollision mit von nativen Anwendungen erstellten Synchronisierungsobjekten zu vermeiden. Sie können Synchronisierungsobjekte von Anwendungen isolieren, die nicht in demselben virtuellen Namensraum ausgeführt werden. Wenn zwei Anwendungen denselben Sandbox-Pfad gemeinsam nutzen, haben die Anwendungen denselben Namensraum für isolierte Synchronisierungsobjekte. Wenn zwei Anwendungen denselben Sandbox-Namen, aber verschiedene Sandbox-Pfade haben, haben die Anwendungen unterschiedliche Namensräume.

Der Parameter `IsolatedSynchronizationObjects` akzeptiert eine Liste mit durch Semikolon (;) getrennten Einträgen. Jeder Eintrag kann die Platzhalter Sternchen (*) und Fragezeichen (?) verwenden, damit er auf variable Muster zutrifft.

Beispiele

Sie können zwei Synchronisierungsobjekte isolieren, ein Objekt mit `outlook` im Namen zuordnen und ein Objekt mit dem exakten Namen `My Shared Object (Mein gemeinsam genutztes Objekt)` zuordnen.

```
[BuildOptions]
IsolatedSynchronizationObjects=*outlook*;My Shared Object (Mein gemeinsam genutztes Objekt)
```

ObjectTypes

Der Parameter `ObjectTypes` gibt eine Liste mit virtuellen COM-Objekttypen an, die für andere Anwendungen in der physischen Umgebung sichtbar sind. Sie können Skripte wie z.B. VBScripts verwenden, um Objekte aufzurufen, die gekapselte Anwendungen starten.

Ein Objekttyp kann jeweils nur für eine native oder eine virtuelle Anwendung registriert werden. Wenn Sie Office 2003 auf dem nativen Rechner installieren und ein virtuelles Office 2007-Paket verwenden möchten, müssen Sie auswählen, ob die virtuelle oder die native Anwendung die Objekttypen verarbeiten soll.

Wenn Sie möchten, dass das virtuelle Office 2007 die Objekttypen verarbeitet, können Sie die Einstellung `ObjectTypes` in der `Package.ini`-Datei belassen, das Paket erstellen und es mithilfe des Dienstprogramms `thinreg.exe` registrieren. Wenn Sie möchten, dass das native Office 2003 die Objekttypen verarbeitet, müssen Sie die Einstellung `ObjectTypes` aus der `Package.ini`-Datei entfernen, bevor Sie das Paket erstellen und registrieren. Sie können keine zufälligen Einträge zum `ObjectTypes`-Parameter hinzufügen. Sie können nur Einträge entfernen, die durch den Setup Capture-Prozess generiert wurden.

Beispiele

Wenn ein Skript oder eine native Anwendung ein COM-Objekt `Excel.Application` oder andere COM-Objekte erstellt, die im Parameter `ObjectTypes` aufgelistet sind, startet ThinApp das virtuelle Paket.

```
[Microsoft Office Excel 2007.exe]
ObjectTypes=Excel.Application;Excel.Application.12;Excel.Chart;
               Excel.Macrosheet;Excel.Sheet; Excel.Workspace
```

SandboxCOMObjects

Der Parameter `SandboxCOMObjects` weist darauf hin, ob Anwendungen in der physischen Umgebung auf COM-Objekte zugreifen können, die bei Laufzeit von der virtuellen Anwendung registriert werden.

Beispiele

Sie können verhindern, dass native Anwendungen in der physischen Umgebung auf COM-Objekte zugreifen, die von der virtuellen Anwendung registriert werden. ThinApp legt in der Sandbox die COM-Objekte ab, die von der virtuellen Anwendung registriert werden.

```
SandboxCOMObjects=1
```

Sie können COM-Objekte sichtbar machen, die von der virtuellen Anwendung außerhalb der Sandbox registriert werden.

```
SandboxCOMObjects=0
```

VirtualizeExternalOutOfProcessCOM

Der Parameter `VirtualizeExternalOutOfProcessCOM` steuert, ob externe COM-Objekte außerhalb der Prozesse in der virtuellen Umgebung ausgeführt werden können.

Gekapselte Anwendungen können COM-Objekte vom Hostsystem erstellen sowie COM-Objekte, die von ThinApp in der virtuellen Umgebung registriert werden.

Der Parameter `VirtualizeExternalOutOfProcessCOM` legt fest, wie mit COM-Objekten außerhalb von Prozessen verfahren werden soll, die nicht zu einem ThinApp-Paket gehören und die nicht in der virtuellen Registrierung eingetragen sind. ThinApp führt externe COM-Objekte außerhalb von Prozessen in der virtuellen Umgebung aus, um sicherzustellen, dass die COM-Objekte den Hostcomputer nicht modifizieren können. Falls ein Kompatibilitätsproblem mit einem externen COM-Objekt besteht, das in der virtuellen Umgebung ausgeführt wird, können Sie mit dem Parameter `VirtualizeExternalOutOfProcessCOM` COM-Objekte auf dem Hostsystem erstellen und ausführen. Wenn Sie nur spezifische COM-Objekte außerhalb der virtuellen Umgebung ausführen möchten, können Sie mithilfe des Parameters `ExternalCOMObjects` die CLSID der einzelnen COM-Objekte explizit auflisten.

Beispiele

Sie können alle externen COM-Objekte außerhalb der Prozesse in der physischen Umgebung anstatt in der virtuellen Umgebung ausführen.

```
[BuildOptions]
VirtualizeExternalOutOfProcessCOM=0
```

Verwenden Sie den Standartwert, sodass alle externen COM-Objekte außerhalb der Prozesse in der virtuellen Umgebung ausgeführt werden.

```
[BuildOptions]
VirtualizeExternalOutOfProcessCOM=1
```

Konfigurieren von Speicherplatz

ThinApp-Parameter können verwendet werden, um Dateispeicher zu konfigurieren und virtuelle Laufwerke einzurichten.

CachePath

Der Parameter `CachePath` legt den Pfad zum Cache-Verzeichnis fest, in dem Schriftartdateien und ausführbare Stub-Dateien gespeichert werden. Mit diesem Parameter können Sie erzwingen, dass das Cache-Verzeichnis auf einem anderen Laufwerk gespeichert wird.

Dieser Parameter kann Makros enthalten, beispielsweise `%Local AppData%`, die vor der Benutzung erweitert werden. Wenn der Pfad relativ ist, interpretiert ThinApp den Pfad im Verhältnis zum Verzeichnis, in dem das Paket gespeichert ist.

Mit der Umgebungsvariablen `THINSTALL_CACHE_DIR` können Sie diesen Parameter zur Laufzeit außer Kraft setzen.

Wenn weder der Parameter `CachePath` noch die Umgebungsvariable `THINSTALL_CACHE_DIR` vorhanden ist, verwendet ThinApp einen Standardspeicherort. Der Standardspeicherort richtet sich danach, ob ein `SandboxPath`-Parameter in der `Package.ini`-Datei vorhanden ist. Wenn der `SandboxPath`-Parameter vorhanden und die Pfadeinstellung relativ ist, verwendet `CachePath` standardmäßig denselben Pfad. Wenn die `SandboxPath`-Einstellung vorhanden und die Pfadeinstellung absolut ist, verwendet `CachePath` standardmäßig den Pfad `%Local AppData%\Thinstall\Cache\Stubs`.

Beispiele

Sie können als Cache-Verzeichnis `C:\VirtCache` festlegen.

```
CachePath=C:\VirtCache
```

Wenn das Paket in `C:\VirtApps` gespeichert ist und der Parameter `CachePath` den Wert `Cache` aufweist, lautet das Cache-Verzeichnis `C:\VirtApps\Cache`.

Durch die Verwendung eines USB-Schlüssels kann die Sandbox auf den Schlüssel gezwungen werden. Wenn Sie Pakete im Verzeichnis `\VirtApps` auf dem USB-Schlüssel speichern, können Sie erzwingen, dass das Cache-Verzeichnis im Schlüssel gespeichert wird.

```
CachePath=Sandbox
```

UpgradePath

Der Parameter `UpgradePath` gibt den Speicherort von Informationen und Dateien für Updates an. Application Sync und Integer-Updates verwenden diesen Speicherort.

Standardmäßig speichert das Dienstprogramm Application Sync seine Protokoll- und Cache-Dateien am selben Speicherplatz wie die ausführbare Anwendungsdatei auf dem lokalen Computer. Integer-Updates funktionieren auf ähnliche Weise. Verfügt der Standardspeicherort nur über beschränkten Speicherplatz oder möchten Sie Upgrades von der ausführbaren Anwendungsdatei isolieren, verwenden Sie den Parameter `UpgradePath`, um einen alternativen Speicherort anzugeben. Sie können Umgebungsvariablen im Pfad verwenden. Verwenden Sie keine Ordnermakros.

Wenn das Dienstprogramm Application Sync ein Update von einem Server herunterlädt, speichert es das Update unter einem temporären Namen an dem Speicherort, der unter `UpgradePath` angegeben ist. Beim nächsten Starten der Anwendung benennt ThinApp die temporäre Datei mit einer Dateierweiterung .1 oder .2 um, je nachdem, ob .1 bereits vorhanden ist. ThinApp versucht den Namen mit der Erweiterung .1 in den Originalnamen der Datei zu ändern, der möglicherweise in einem anderen Verzeichnis gespeichert ist. Wenn ThinApp diese Änderung nicht durchführt, behält die Datei die Erweiterung .1 an dem Speicherort von `UpgradePath` bei. Bei der Ausführung der Originalanwendung wird auf diese Datei zugegriffen.

Informationen über das Dienstprogramm Application Sync erhalten Sie unter „Application Sync-Updates“ auf Seite 41.

Beispiele

Anstatt Update-Dateien am Standardspeicherort mit der ausführbaren Anwendungsdatei zu speichern, können Sie ThinApp anweisen, Anwendungs-Updates unter `C:\Programme\MyAppUpgrades` zu erkennen.

```
[BuildOptions]
UpgradePath=C:\Programme\MyAppUpgrades
```

VirtualDrives

Der Parameter `VirtualDrives` gibt zusätzliche Laufwerkbuchstaben an, die bei Laufzeit für die Anwendung verfügbar sind.

Virtuelle Laufwerke sind nützlich, wenn Anwendungen auf hartcodierte Pfade zu Laufwerkbuchstaben zurückgreifen, die auf den Clientcomputern möglicherweise nicht verfügbar sind. Zum Beispiel könnten bestimmte ältere Anwendungen erwarten, dass das Laufwerk `D:` ein CD-ROM-Laufwerk ist und dass Datendateien unter `D:\media` verfügbar sind.

Virtuelle Laufwerke sind nur für diejenigen Anwendungen sichtbar, die in der virtuellen Umgebung ausgeführt werden. Virtuelle Laufwerke wirken sich nicht auf die physische Windows-Umgebung aus. Virtuelle Laufwerke übernehmen Isolationsmodi vom Standardisolationsmodus des Projekts, sofern Sie den Modus nicht spezifisch außer Kraft setzen. Wenn Sie Ihr virtuelles Laufwerk mit dem Wert `Merged (Zusammengeführt)` für den Parameter `IsolationMode` konfigurieren, schlagen sämtliche Schreibvorgänge für das betreffende Laufwerk fehl, wenn dieses nicht auf dem physischen System vorhanden ist.

In einem Projekt werden Informationen über die zum Zeitpunkt der Anwendungskapselung vorhandenen virtuellen Laufwerke aufgelistet. Der Parameter `VirtualDrives` trennt die zu verschiedenen Laufwerkbuchstaben zugewiesenen Informationen durch Semikolon; Parameter für einzelne Laufwerkbuchstaben werden durch Kommata getrennt. Der Parameter `VirtualDrives` enthält die folgenden Informationen:

- `Drive (Laufwerk)` ist ein einzelnes alphabetisches Zeichen von A bis Z.
- `Serial (Seriell)` ist eine achtstellige Hexadezimalzahl.
- `Type (Typ)` ist `FIXED (FEST)`, `REMOVABLE (ENTFERNBAR)`, `CD–ROM` oder `RAMDISK`.
  - `FIXED (FEST)` – Weist auf feste Medien hin.

    Beispiel: eine Festplatte oder ein eingebautes Flash-Laufwerk.
  - `REMOVABLE (ENTFERNBAR)` – Weist auf Wechselmedien hin.

    Beispiel: ein Festplattenlaufwerk, ein Thumb-Laufwerk oder ein Flash-Kartenlesegerät.
  - `CD–ROM` – Weist auf ein CD-ROM-Laufwerk hin.
  - `RAMDISK` – Weist auf eine RAM-Diskette hin.

Beispiele

Der Parameter `VirtualDrives` ist eine einzelne Zeichenfolge, die Informationen für mehrere Laufwerkbuchstaben und optionale Parameter für die betreffenden Laufwerkbuchstaben enthalten kann.

```
VirtualDrives= Drive=A, Serial=12345678, Type=REMOVABLE; Drive=B, Serial=9ABCDEF0, Type=FIXED
```

Beim einfachen Gebrauch wird ein einzelner Buchstabe für ein virtuelles Laufwerk angegeben. Standardmäßig weist ThinApp eine Seriennummer und den Typ `FIXED (FEST)` zum Laufwerk zu.

Sie können als virtuelle Laufwerkbuchstaben die Buchstaben `X`, `D` und `Z` angeben.

```
[BuildOptions]
VirtualDrives=Drive=X, Serial=ff897828, Type=REMOVABLE; Drive=D, Type=CDROM; Drive=Z
```

Laufwerk `X` ist ein Wechseldatenträger mit der Seriennummer ff797828.

Laufwerk `D` ist ein CD-ROM-Laufwerk mit einer zugewiesenen Seriennummer.

Laufwerk `Z` ist ein `FIXED (FESTES)` Laufwerk mit einer zugewiesenen Seriennummer.

Ändern der Isolationseinstellungen für ein virtuelles Laufwerk

Eventuell müssen Sie mithilfe der `##Attributes.ini`-Datei den Isolationsmodus für ein virtuelles Laufwerk ändern.

So geben Sie den Isolationsmodus für ein virtuelles Laufwerk an

1 Fügen Sie den Ordner `%Drive_X%` zu Ihrem ThinApp-Projekt hinzu.

2 Fügen Sie in dem neuen Verzeichnis die `##Attributes.ini`-Datei hinzu, um den Isolationsmodus für den Laufwerkbuchstaben anzugeben.

Konfigurieren von Prozessen und Diensten

Zum Konfigurieren von Prozessen und Diensten werden Parameter verwendet, die den Schreibzugriff für native Prozesse oder für den Start und das Beenden virtueller Dienste angeben.

AllowExternalProcessModifications

Der Parameter `AllowExternalProcessModifications` legt fest, ob die gekapselten Anwendungen in einen nativen Prozess schreiben können. Zum Beispiel kann es vorkommen, dass Sie eine Spracherkennungsanwendung kapseln, die sich selbst in native Anwendungen einfügen muss, um Text in Sprache umzuwandeln.

Wenn ThinApp die Selbsteinspeisung einer gekapselten Anwendung in eine native Anwendung blockiert, generiert Protokoll-Monitor Ablaufverfolgungsprotokolle, die auf den Parameter `AllowExternalProcessModifications` verweisen.

Beispiele

Der Standardwert für den Parameter `AllowExternalProcessModifications` blockiert alle Versuche der gekapselten Anwendung, sich selbst in eine native Anwendung einzuspeisen. Die gekapselte Anwendung kann sich immer noch in virtuelle Anwendungen einfügen, die in derselben Sandbox ausgeführt werden. ThinApp zeigt den Standardparameter nicht in der `Package.ini`-Datei an.

```
[BuildOptions]
AllowExternalProcessModifications=0
```

Sie können den Parameter `AllowExternalProcessModifications` manuell mit dem Wert 1 in die `Package.ini`-Datei einfügen, um das Standardblockierverhalten zu verhindern.

```
[BuildOptions]
AllowExternalProcessModifications=0
```

AllowUnsupportedExternalChildProcesses

Der Parameter `AllowUnsupportedExternalChildProcesses` gibt an, ob die virtualisierte Anwendung an der Erstellung eines untergeordneten 64-Bit-Prozesses gehindert werden soll. Sie können untergeordnete 64-Bit-Prozesse in der physischen Umgebung anstatt in der virtuellen Umgebung erstellen.

Wenn Sie keinen Wert angeben, wird das Standardverhalten angewandt, das nicht unterstützte externe Prozesse ermöglicht.

Beispiele

Die Standardeinstellung des Parameters `AllowUnsupportedExternalChildProcesses` bewirkt, dass ThinApp 64-Bit-Anwendungen in der physischen Umgebung ausführt. Sie können Aufgaben von untergeordneten 64-Bit-Prozessen ausführen, die auf 64-Bit-Systemen ausgeführt werden. Die Ausführung der Druckwarteschlange ist ein Beispiel für eine Aufgabe eines untergeordneten 64-Bit-Prozesses.

```
[BuildOptions]
AllowUnsupportedExternalChildProcesses=1
```

Sie können die Generierung von untergeordneten 64-Bit-Prozessen außerhalb der virtuellen Umgebung durch ThinApp blockieren.

```
AllowUnsupportedExternalChildProcesses=0
```

AutoShutdownServices

Der Parameter `AutoShutdownServices` steuert, ob virtuelle Dienste beim Beenden des letzten dienstfremden Prozesses heruntergefahren werden.

Mit dem Standardverhalten werden virtuelle Dienste heruntergefahren, wenn der letzte untergeordnete dienstfremde Prozess beendet wird. Der Parameter `AutoShutdownServices` weist ThinApp an, virtuelle Dienste weiter auszuführen, selbst wenn alle anderen Prozesse beendet werden. Der Parameter wirkt sich nicht auf Dienste außerhalb des virtuellen Kontexts aus.

Beispiele

Sie können virtuelle Dienste weiter ausführen lassen, wenn die Anwendung beendet wird.

```
[BuildOptions]
AutoShutdownServices=0
```

Sie können virtuelle Dienste beenden, wenn die letzte dienstfremde Anwendung beendet wird. Dies ist das Standardverhalten.

```
[BuildOptions]
AutoShutdownServices=1
```

AutoStartServices

Der Parameter `AutoStartServices` steuert, ob virtuelle Dienste gestartet werden, wenn die erste Anwendung gestartet wird.

Mit dem Standardverhalten werden virtuelle Dienste gestartet, die mit dem Starttyp **Automatisch (Automatic)** installiert wurden. Die virtuellen Dienste werden gestartet, wenn der Benutzer den ersten übergeordneten Prozess ausführt. Mit dem Parameter `AutoStartServices` können Sie das automatische Starten von virtuellen Diensten deaktivieren.

Beispiele

Sie können das Starten virtueller Dienste verhindern.

```
[BuildOptions]
AutoStartServices=0
```

Sie können virtuelle Dienste starten, wenn der erste Prozess gestartet wird. Dies ist das Standardverhalten.

```
[BuildOptions]
AutoStartServices=1
```

ChildProcessEnvironmentDefault

Der Parameter `ChildProcessEnvironmentDefault` legt fest, ob ThinApp alle untergeordneten Prozesse in der virtuellen Umgebung ausführt.

Mit dem Parameter `ChildProcessEnvironmentExceptions` können Sie spezifische Ausnahmen erstellen. Siehe „ChildProcessEnvironmentExceptions“ auf Seite 81.

Beispiele

Der Standardeintrag erstellt alle untergeordneten Prozesse in der virtuellen Umgebung.

```
[BuildOptions]
ChildProcessEnvironmentDefault=Virtual
```

Der Wert `External (Extern)` erstellt untergeordnete Prozesse außerhalb der virtuellen Umgebung.

```
[BuildOptions]
ChildProcessEnvironmentDefault=External (Extern)
```

ChildProcessEnvironmentExceptions

Der Parameter `ChildProcessEnvironmentExceptions` weist auf Ausnahmen für den `ChildProcessEnvironmentDefault`-Parameter hin.

Wenn für den Parameter `ChildProcessEnvironmentDefault` der Wert `Virtual (Virtuell)` festgelegt wird, listet der Parameter `ChildProcessEnvironmentExceptions` die Anwendungen auf, die außerhalb der virtuellen Umgebung ausgeführt werden. Wenn für den Parameter `ChildProcessEnvironmentDefault` der Wert `External (Extern)` festgelegt wird, listet der Parameter `ChildProcessEnvironmentExceptions` die Anwendungen auf, die innerhalb der virtuellen Umgebung ausgeführt werden.

Beispiele

Sie können Ausnahmen für die Ausführung von untergeordneten Prozessen in der virtuellen Umgebung angeben. Wenn die virtuelle Anwendung einen untergeordneten Prozess `notepad.exe` startet, wird der untergeordnete Prozess außerhalb der virtuellen Umgebung ausgeführt.

```
[BuildOptions]
ChildProcessEnvironmentExceptions=AcroRd.exe;notepad.exe
ChildProcessEnvironmentDefault=Virtual (Virtuell)
```

Konfigurieren von Datei- und Blockgrößen

ThinApp-Parameter können verwendet werden, um Datei- und Blockgrößen von Anwendungen zu komprimieren.

BlockSize

Der Parameter `BlockSize` steuert die Größe der Komprimierungsblöcke, wenn ThinApp Dateien für einen Build komprimiert.

Durch eine größere Blockgröße kann eine höhere Komprimierung erzielt werden. Allerdings kann die Leistung durch eine höhere Blockgröße sinken. Dies hat folgende Gründe:

- Der Build-Prozess verlangsamt sich bei größeren Blockgrößen.
- Die Startdauer und die Lesevorgänge für Anwendungen verlangsamen sich bei größeren Blockgrößen.
- Bei Laufzeit ist mehr Arbeitsspeicher erforderlich, wenn größere Blockgrößen verwendet werden.

Sie können den Parameter `BlockSize` in der `Package.ini`-Datei angeben. Die Blockgröße wird dann zum Standardwert für alle Dateien in dem Projekt, sofern nichts anderes angegeben ist. Oder Sie können den Parameter in der `##Attributes.ini`-Datei angeben. Die dort angegebene Blockgröße hat Vorrang gegenüber der Blockgröße für das vorhandene Verzeichnis und sämtliche Unterverzeichnisse. Sie können verschiedene Blockgrößen für unterschiedliche Verzeichnisse innerhalb desselben Projekts verwenden.

Beispiele

Sie können als Standardblockgröße 64 KB festlegen.

```
[Compression]
BlockSize=64k
```

Sie können andere Blockgrößen verwenden.

```
BlockSize=128k
BlockSize=256k
BlockSize=512k
BlockSize=1M
```

CompressionType

Der Parameter `CompressionType` legt den Komprimierungswert auf `None (Ohne)` oder `Fast (Schnell)` fest.

`None (Ohne)` ist der Standardwert, wenn Sie eine Anwendung kapseln. Dieser Wert ist nützlich für die schnelle Erstellung Ihrer Anwendung zu Testzwecken. Durch Vermeidung der Komprimierung wird die Startdauer der Anwendung auf älteren Computern verbessert oder wenn Sie die Anwendung mehrere Male starten und für die Bereitstellung der Daten für die einzelnen Startvorgänge auf den Windows-Festplatten-Cache angewiesen sind.

Die schnelle Komprimierung hat eine schnelle Dekomprimierungsrate und wirkt sich nur unwesentlich auf die Startdauer der Anwendung und auf den Arbeitsspeicherverbrauch bei Laufzeit aus. Durch die schnelle Komprimierung werden ähnliche Komprimierungsverhältnisse erzielt wie mit dem ZIP-Algorithmus.

In Tabelle A-1 werden Beispiele für Komprimierungsverhältnisse und Startdauer für ein Microsoft Office 2003-Paket aufgelistet, das von einer lokalen Festplatte ausgeführt wird.

Tabelle A-1. Beispiele für Komprimierungsverhältnisse und Startdauer

Komprimierungstyp	None (Ohne)	Fast (Schnell)
Größe	448.616 KB	257.373 KB
Komprimierungsverhältnis	100%	57%
Startdauer (erste Ausführung)	6 Sekunden	6 Sekunde
Startdauer (zweite Ausführung)	0,1 Sekunden	1 Sekunde
Build-Zeit (erste Erstellung)	3 Minuten	19 Minuten
Build-Zeit (zweite Erstellung)	2 Minuten	1,2 Minuten

Sie können den Parameter `CompressionType` in der `Package.ini`-Datei angeben. Der Komprimierungstyp wird dann zum Standardwert für alle Dateien in dem Projekt, sofern nichts anderes angegeben ist. Und Sie können den Parameter in der `##Attributes.ini`-Datei angeben. Der dort angegebene Komprimierungstyp hat Vorrang gegenüber dem Komprimierungsalgorithmus für das vorhandene Verzeichnis und sämtliche Unterverzeichnisse. Sie können verschiedene Komprimierungsalgorithmen für unterschiedliche Verzeichnisse innerhalb desselben Projekts verwenden.

Beispiele

Sie können eine Komprimierung verhindern, um eine schnelle Build- und Ladezeit zu erzielen. Dies ist das Standardverhalten.

```
[Compression]
CompressionType=None (Ohne)
```

Sie können die schnelle Komprimierung für eine langsame Build-Zeit und für eine schnelle Ladezeit verwenden.

```
[Compression]
CompressionType=Fast (Schnell)
```

Konfigurieren von Symbolen

ThinApp-Parameter können verwendet werden, um Symbole hinzuzufügen oder zu entfernen.

Icon

Der Parameter `Icon (Symbol)` gibt die Symboldatei an, die für die generierte ausführbare Datei verwendet werden soll.

Standardmäßig verwendet jede generierte Anwendung das Hauptgruppensymbol von ihrer ausführbaren Quelldatei und die spezifische Symbolressource, auf die das Gruppensymbol verweist. Sie können die Verwendung eines anderen Symbols für eine `.ico`-Datei oder ausführbare Datei angeben.

Beispiele

Sie können einen `NULL`-Wert angeben, um eine ausführbare Datei ohne Symbole zu generieren. Verwenden Sie keinen `NULL`-Wert, wenn Sie die Dateityprichtlinie verwenden. Die Abbildung der ausführbaren Datei weist jedem Dateityp ein Symbol zu.

```
[myapp.exe]
Source=%ProgramFilesDir%\myapp\app.exe
Icon=NULL
```

Sie können das Anwendungssymbol mithilfe einer ausführbaren Datei angeben, die sich von der ausführbaren `Source`-Datei (Quelldatei) unterscheidet.

```
[myapp.exe]
Source=%ProgramFilesDir%\myapp\app.exe
Icon=%ProgramFilesDir%\myapp\app2.exe
```

Durch Anhängen von `,1` `,2` am Ende des Symbolpfades können Sie angeben, welcher Satz verwendet werden soll.

```
[myapp.exe]
Source=%ProgramFilesDir%\myapp\app.exe
Icon=%ProgramFilesDir%\myapp\app2.exe,1
```

Mithilfe einer `.ico`-Datei können Sie das Anwendungssymbol angeben.

```
[myapp.exe]
Source=%ProgramFilesDir%\myapp\app.exe
Icon=%ProgramFilesDir%\myap\myicon.ico
```

RetainAllIcons

Der Parameter `RetainAllIcons` behält sämtliche Originalsymbole der ausführbaren Quelldatei in der gekapselten ausführbaren Datei bei.

Standardmäßig erstellt das Dienstprogramm `tlink.exe` eine neue ausführbare Datei mithilfe einer ausführbaren Quelldatei. Zur Verringerung des benötigten Festplattenspeichers enthält die Abbildung der neuen ausführbaren Datei nur Symbole, die von der System-Shell aus angezeigt werden können. Das Paket enthält alle anderen Symbole. Auf die Symbole kann von der Anwendung während ihrer Ausführung weiter zugegriffen werden. Die Symbole, auf die das System zugreifen kann, belegen mehr Festplattenspeicher, weil ThinApp die Symbole nicht komprimieren kann. Möglicherweise möchten Sie, dass alle Originalsymbole der Anwendung für die System-Shell sichtbar sind.

Beispiele

Sie können das Dienstprogramm `tlink.exe` anweisen, alle Originalsymbole der Anwendung beizubehalten.

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
RetainAllIcons=1
```

Beim Standardverhalten werden unbenutzte Symbole aus dem für die physische Umgebung sichtbaren Teil der ausführbaren Datei entfernt.

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
RetainAllIcons=0
```

Konfigurieren der Protokollierung

ThinApp-Parameter können verwendet werden, um Protokollierungsaktivitäten zu verhindern, oder den Speicherort der Protokolldateien anzupassen.

DisableTracing

Der Parameter `DisableTracing` verhindert die Generierung einer `.trace`-Datei beim Ausführen von Protokoll-Monitor. Protokoll-Monitor erzeugt `.trace`-Dateien zur Problembehandlung.

Die Generierung einer `.trace`-Datei zu deaktivieren, kann aus folgenden Gründen sinnvoll sein:

- Um den Ausführungsverlauf aus Sicherheitsgründen zu verbergen.
- In einer Testumgebung kann es erforderlich sein, die Ablaufverfolgung für bestimmte Anwendungen auszuschalten, von denen Sie wissen, dass sie ordnungsgemäß funktionieren. Durch die Erzeugung überflüssiger `.trace`-Dateien werden Festplattenspeicher und CPU-Zeit vergeudet.

Beispiele

Sie können die Erstellung einer `.trace`-Datei durch eine Anwendung verhindern, selbst wenn Sie Protokoll-Monitor ausführen.

```
[BuildOptions]
DisableTracing=1
```

Beim Standardverhalten wird die Generierung einer `.trace`-Datei in Protokoll-Monitor unterstützt.

```
[BuildOptions]
DisableTracing=0
```

LogPath

Der Parameter `LogPath` legt den Speicherort fest, an dem `.trace`-Dateien während der Protokollierungsaktivität gespeichert werden sollen. Der Standardspeicherort ist dasselbe Verzeichnis, in dem auch die ausführbare Anwendungsdatei gespeichert wird. Sie können den Standardspeicherort ändern, um ein Verzeichnis mit größerem Speicherplatz zu finden oder, um die Protokolle von einem USB-Gerät auf den Clientcomputer umzuleiten.

Beispiele

Sie können ThinApp so anweisen, dass die Protokolldateien in `c:\ThinappLogs` gespeichert werden.

```
[BuildOptions]
LogPath=C:\ThinappLogs
```

Im Gegensatz zu den meisten Pfaden in ThinApp darf der Protokollpfad keine Makros wie `%AppData%` oder `%Temp%` enthalten.

Konfigurieren von Versionen

ThinApp-Parameter bieten Informationen über die Versionen der ausführbaren Anwendungsdateien und ThinApp.

CapturedUsingVersion

Der Parameter `CapturedUsingVersion` weist auf die Version des Setup Capture-Assistenten hin, die bei der Kapselung der Anwendung verwendet wird.

Beispiele

Sie brauchen diesen Parameter nicht anzupassen.

```
[BuildOptions]
CapturedUsingVersion=4.0.0–2200
```

StripVersionInfo

Der Parameter `StripVersionInfo` entfernt alle Versionsinformationen aus der ausführbaren Quelldatei, wenn ThinApp die Anwendung erstellt.

Die Versionsinformationen für ausführbare Dateien befinden sich in den Windows-Eigenschaften. Die Informationen über Eigenschaften umfassen Informationen zum Urheberrecht, zu Marken und zur Versionsnummer. Standardmäßig kopiert ThinApp alle Versionsinformationen von der ausführbaren Quelldatei. Der Parameter `StripVersionInfo` entfernt die Versionsinformationen aus der gekapselten Anwendung.

Beispiele

Sie können eine Zielanwendung ohne Versionsinformationen generieren.

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
StripVersionInfo=1
```

Version.XXXX

Der Parameter `Version.XXXX` überschreibt Versionszeichenfolgen der ausführbaren Datei oder fügt neue Versionszeichenfolgen hinzu.

ThinApp kopiert Versionsressourcen von der ausführbaren Originaldatei. Sie können die Zeichenfolgen mit den Versionsressourcen überschreiben und neue hinzufügen. Dies geschieht mit der Einstellung `Version.<Zeichenfolge_Name>=<Zeichenfolge_Wert>`.

Beispiele

Sie können als Produktnamenwert für die Version den Wert `My New Product Name (Mein neuer Produktname)` festlegen.

```
[<Anw>.exe]
Version.ProductName=My New Product Name (Mein neuer Produktname)
Version.Description=This Product is great! (Dieses Produkt ist fantastisch!)
```

Konfigurieren von lokalen Informationen

ThinApp-Parameter, die lokale Informationen anzeigen, erfordern keine Änderungen.

AnsiCodePage

Der Parameter `AnsiCodePage` verwendet einen numerischen Wert, um das Gebietsschema anzugeben, in der die Anwendung gekapselt wurde. ThinApp verwendet diesen Wert, um Multibyte-Zeichenfolgen zu übersetzen.

Beispiele

Der Kapselungsprozess generiert den `AnsiCodePage`-Wert.

```
[BuildOptions]
AnsiCodePage=1252
```

LocaleIdentifier

Der Parameter `LocaleIdentifier` zeigt eine numerische ID für das Gebietsschema an. Der Wert sucht die geeigneten Sprachressourcen von der Anwendung.

Beispiele

1033 ist die Gebietsschema-ID für eine Anwendung in englischer Sprache.

```
[BuildOptions]
LocaleIdentifier=1033
```

LocaleName

Der Parameter `LocaleName` zeigt den Namen für das Gebietsschema an, wenn Sie eine Anwendung in Microsoft Vista kapseln.

Beispiele

ThinApp kann einen Namen für das japanische Gebietsschema generieren.

```
[BuildOptions]
LocaleName=ja-JP
```

Konfigurieren von einzelnen Anwendungen

Einstiegspunktspezifische Parameter fallen unter die Sektionen `[<Anwendung>.exe]` der `Package.ini`-Datei. Zum Beispiel wirken sich Einträge unter `[Adobe Reader 8.exe]` für eine Adobe Reader-Anwendung auf Bereiche wie Befehlszeilenargumente und Anwendungsverknüpfungen aus.

CommandLine

Der Parameter `CommandLine` gibt die Befehlszeilenargumente aus, die eine verknüpfte ausführbare Datei starten. Während der Parameter `Source (Quelle)` den Pfad zur verknüpften ausführbaren Datei angibt, gibt der Parameter `CommandLine` die Datei mit den erforderlichen Optionen oder Parametern an.

Die Optionen und Parameter folgen dem Basisanwendungsnamen. Verwenden Sie, abhängig von der Anwendung, / oder – vor der Option oder dem Parameter. Verwenden Sie Ordnermakros für die Pfadbenennungskonventionen.

Enthält die **Start**menüverknüpfung der Anwendung Befehlszeilenoptionen, bestimmt ThinApp den Wert des Parameters `CommandLine`, basierend auf diesen Optionen. In seltenen Fällen der Problembehandlung müssen Sie diesen Parameter gegebenenfalls ändern.

Beispiele

Verwenden Sie das Format `/<Option> <Parameter>` für Befehlszeilenargumente.

```
[<Anw>.exe]
Source=%ProgramFilesDir%\<Basis_Anw>\<Anw>.exe
Shortcut=<Primärer_Datenbehälter>.exe
CommandLine="%ProgramFilesDir%\<Basis_Anw>\<Anw>.exe" /<Option> <Parameter>
```

ThinApp kann einen `CommandLine`-Eintrag basierend auf der **Start**menüverknüpfung `"C:\Programme\Mozilla Firefox\firefox.exe" –safe–mode` erstellen.

```
CommandLine="C:\Program Files\Mozilla Firefox\firefox.exe" –safe–mode
```

Disabled

Der Parameter `Disabled (Deaktiviert)` zeigt an, dass das Build-Ziel einer Anwendung nur ein Platzhalter ist, und verhindert, dass ThinApp die ausführbare Datei im `/bin`-Verzeichnis generiert. Dieser Parameter ist nützlich, wenn Sie keinen speziellen Einstiegspunkt auswählen, während der Setup Capture-Assistent ausgeführt wird, doch zu einem späteren Zeitpunkt entscheiden, dass Sie eine ausführbare Datei für diesen Einstiegspunkt generieren möchten.

Wenn Sie bei der Kapselung der Anwendung weder `cmd.exe`, `regedit.exe` noch `iexplore.exe` als Einstiegspunkt auswählen und Sie die Umgebung debuggen oder Fehler in der Umgebung beheben müssen, können Sie den Parameter `Disabled (Deaktiviert)` auf den Wert 0 setzen und das Projekt neu erstellen, um diese Einstiegspunkte zu generieren. Informationen zur Problembehandlung bei Einstiegspunkten erhalten Sie unter „Festlegen von Anwendungsverknüpfungen und Nachverfolgungsnamen“ auf Seite 17.

Beispiele

Sie können die Generierung der ausführbaren Datei für die Anwendung während des Build-Prozesses verhindern. ThinApp verwendet diese Einstellung für die Einstiegspunkte, die Sie nicht auswählen, während der Setup Capture-Assistent ausgeführt wird.

```
[app.exe]
Source=%ProgramFilesDir%\<Meine_Anw>\<Anw>.exe
Disabled=1
```

Sie können den Parameter `Disabled (Deaktiviert)` auf den Wert 0 setzen oder die Zeile für die Generierung der ausführbaren Datei für die Anwendung entfernen.

```
[app.exe]
Source=%ProgramFilesDir%\<Meine_Anw>\<Anw>.exe
Disabled=0
```

ReadOnlyData

Der Parameter `ReadOnlyData` gibt den Namen der schreibgeschützten virtuellen Registrierungsdatei an, die bei der Erstellung der Anwendung erstellt wird.

Ändern Sie den Wert dieses Parameters nicht. Die `Package.ini`-Datei zeigt diesen Parameter an, falls Sie die primäre Datencontainerdatei lokalisieren müssen. Der Parameter `ReadOnlyData` in einer anwendungsspezifischen Sektion der `Package.ini`-Datei bestimmt die primäre Datencontainerdatei.

Ist die primäre Datencontainerdatei kleiner als 200 MB, wird der Container in einer ausführbaren Datei mit Einstiegspunkt gespeichert. Ist die primäre Datencontainerdatei größer als 200 MB, speichert ThinApp den Container als `.dat`-Datei, die nicht als Einstiegspunkt für die Anwendung dienen kann.

Beispiele

Der Standard- und erforderliche Wert gibt `Package.ro.tvr` als Namen der virtuellen Registrierungsdatei an.

```
ReadOnlyData=bin\Package.ro.tvr
```

ReserveExtraAddressSpace

Der Parameter `ReserveExtraAddressSpace` gibt die Größe des zusätzlichen Adressbereichs an, der für die gekapselte ausführbare Datei reserviert werden soll.

Das Dienstprogramm `tlink.exe` legt das `SizeOfImage`-Feld für Windows in der generierten ausführbaren Datei auf Basis des `SizeOfImage`-Feldes der ausführbaren Quelldatei fest. Das Windows-Ladeprogramm verwendet das `SizeOfImage`-Feld, um zu bestimmen, wie groß der virtuelle Adressbereich sein muss, der für die ausführbare Datei reserviert werden soll. Wenn Sie ein Paket auf Basis einer ausführbaren Quelldatei erstellen, die nicht in dem betreffenden Paket enthalten ist, können Sie virtuellen Adressbereich durch Angabe des Parameters `ReserveExtraAddressSpace` reservieren. Als Wert wird die Anzahl der zu reservierenden Bytes angegeben. Sie können `K` an die Zahl anhängen, um Kilobytes anzugeben, oder `M` zur Angabe von Megabytes. Der Standardwert `0` gibt den zu reservierenden Adressbereich an.

Beispiele

Sie können das Windows-Ladeprogramm anweisen, einen Adressbereich von 512 KB zu reservieren.

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
ReserveExtraAddressSpace=512K
```

Beim Standardverhalten wird kein zusätzlicher Adressbereich reserviert.

```
[app.exe]
Source=%ProgramFilesDir%\myapp\app.exe
ReserveExtraAddressSpace=0
```

Shortcut

Der Parameter `Shortcut (Verknüpfung)` verweist eine ausführbare Datei auf einen primären Datencontainer, der das virtuelle Dateisystem und die virtuelle Registrierung enthält. Ein primärer Datencontainer kann in der `Package.ini`-Datei von anderen Einstiegspunkten unterschieden werden, da der primäre Datencontainer den `ReadOnlyData`-Eintrag enthält und die anderen Einstiegspunkte den `Shortcut (Verknüpfung)`-Eintrag enthalten.

Die verknüpfte ausführbare Datei muss im selben Verzeichnis gespeichert sein wie die primäre Datencontainerdatei, damit die Anwendung gestartet werden kann. Informationen über den primären Datencontainer erhalten Sie unter „ReadOnlyData“ auf Seite 87.

Ändern Sie den Wert des Parameters `Shortcut (Verknüpfung)`nicht. ThinApp erkennt den primären Datencontainer während des Kapselungsprozesses.

Beispiele

ThinApp kann die verknüpfte ausführbare Datei `AcroRd32.exe` auf die primäre Datencontainerdatei `Adobe Reader 8.exe` verweisen.

```
[AcroRd32.exe]
Shortcut=Adobe Reader 8.exe
Source=%ProgramFilesDir%\Adobe\Reader 8.0\Reader\AcroRd32.exe
```

ThinApp kann die verknüpfte ausführbare Datei `Microsoft Office Word 2007.exe` auf die primäre Datencontainerdatei `Microsoft Office Enterprise 2007.dat` verweisen.

```
[Microsoft Office Word 2007.exe]
Source=%ProgramFilesDir%\Microsoft Office\Office12\WINWORD.EXE
Shortcut=Microsoft Office Enterprise 2007.dat
```

Shortcuts

Der Parameter `Shortcuts (Verknüpfungen)` listet die Speicherorte auf, an denen das Dienstprogramm `thinreg.exe` eine Verknüpfung zu einer virtuellen Anwendung erstellt. Sie können die Einträge durch Semikolon trennen. Jeder Eintrag kann Ordnermakros enthalten.

Der Kapselungsprozess bestimmt `Shortcuts (Verknüpfungen)`-Einträge basierend auf den Verknüpfungen, die das Installationsprogramm der Anwendung implementiert. Verwenden Sie zum Trennen der Einträge ein Semikolon, wenn Sie Speicherorte für Verknüpfungen hinzufügen.

MSI-Dateien verwenden den Parameter `Shortcuts (Verknüpfungen)`, um die Verknüpfungen zu bestimmen, die erstellt werden sollen.

Beispiele

Sie können eine Verknüpfung im Microsoft Office-Ordner des **Start**menüs zur Anwendung Microsoft Word 2003 erstellen.

```
[Microsoft Office Word 2003.exe]
ReadOnlyData=bin\Package.ro.tvr
Source=%ProgramFilesDir%\Microsoft Office\OFFICE11\WINWORD.EXE
Shortcuts=%Programs%\Microsoft Office
```

Source

Der Parameter `Source (Quelle)` verweist auf die ausführbare Datei, die ThinApp lädt, wenn Sie eine verknüpfte ausführbare Datei verwenden. Der Parameter liefert den Pfad zur ausführbaren Datei im virtuellen oder physischen Dateisystem. Kann ThinApp die ausführbare Quelldatei nicht im virtuellen Dateisystem lokalisieren, durchsucht ThinApp das physische Dateisystem. Wenn Sie beispielsweise den nativen Internet Explorer aus der virtuellen Umgebung verwenden, lädt ThinApp die ausführbare Quelldatei aus dem physischen Dateisystem.

ThinApp gibt die Quelle für die einzelnen ausführbaren Dateien an. Wenn eine Anwendungssuite drei Benutzereinstiegspunkte hat, zum Beispiel `Winword.exe`, `Powerpnt.exe` und `Excel.exe`, werden in der `Package.ini`-Datei drei Anwendungseinträge aufgelistet. Jeder Eintrag besitzt einen eindeutigen Quelleintrag.

Der Quellparameter und das `/bin`-Verzeichnis im Projekt stehen nicht zueinander in Beziehung. Das `/bin`-Verzeichnis speichert die generierte ausführbare Datei und der `Source (Quell)`-Pfad führt zur installierten ausführbaren Datei, die im schreibgeschützten virtuellen Dateisystem gespeichert ist.

Ändern Sie den `Source (Quell)`-Pfad nicht. Der Kapselungsprozess bestimmt den Pfad basierend darauf, wo das Installationsprogramm die ausführbare Datei im physischen Dateisystem des Computers, mit dem die Kapselung ausgeführt wurde, speichert. ThinApp erstellt einen Pfad für das virtuelle Dateisystem basierend auf dem physischen Dateisystempfad.

Beispiele

ThinApp kann unter `C:\Programme\<Basis_Anw>\<Anw>.exe` einen Einstiegspunkt für eine Anwendung erstellen.

```
[<Anw>.exe]
Source=%ProgramFilesDir%\<Basis_Anw>\<Anw>.exe
```

WorkingDirectory

Der Parameter `WorkingDirectory` legt das aktuelle Arbeitsverzeichnis vor dem Starten der Anwendung fest. Das Arbeitsverzeichnis ist der erste Ort, an dem eine Anwendung nach Dateien sucht und Dateien speichert.

ThinApp schließt diesen Parameter nicht standardmäßig in die `Package.ini`-Datei ein, da ThinApp davon ausgeht, dass das Arbeitsverzeichnis dasjenige Verzeichnis ist, in dem sich die ausführbare Datei befindet. Der typische Speicherort in einer ThinApp-Umgebung ist auf dem Desktop des Arbeitscomputers.

Sie können das Arbeitsverzeichnis für einzelne Anwendungen festlegen. Das Arbeitsverzeichnis kann – abhängig von der Isolationsmoduseinstellung – in einem virtuellen Dateisystem, in der Sandbox oder im physischen System vorhanden sein. Sie können Ordnermakros für die Pfadbenennungskonventionen verwenden.

Der Parameter `WorkingDirectory` legt den anfänglichen Wert des Arbeitsverzeichnisses fest, doch das Verzeichnis ist dynamisch, wenn Sie zu anderen Speicherorten navigieren.

Beispiele

Sie können das Arbeitsverzeichnis für eine Anwendung auf einem USB-Laufwerk vom USB-Standardspeicherort zum Verzeichnis `Eigene Dateien` auf dem Desktop ändern.

```
[<Anw>.exe]
WorkingDirectory=%Personal%
```

Der Speicherort des Verzeichnisses `Eigene Dateien` hängt von der Isolationsmoduseinstellung ab. Verwenden Sie die Isolationsmoduseinstellung `Merged (Zusammengeführt)`, wenn Sie das Arbeitsverzeichnis dem Verzeichnis `Eigene Dateien` auf dem physischen System zuordnen möchten. Wenn Sie das Arbeitsverzeichnis der Sandbox auf dem lokalen Computer zuordnen möchten, verwenden Sie den Isolationsmodus `WriteCopy` oder `Full`. Siehe „DirectoryIsolationMode“ auf Seite 68.

Konfigurieren von abhängigen Anwendungen mit Application Link

Das Dienstprogramm Application Link unterteilt gemeinsam genutzte Komponenten oder abhängige Anwendungen in separate Pakete. In der `Package.ini`-Datei können Sie mithilfe der Einträge `OptionalAppLinks` und `RequiredAppLinks` dynamische Kombinationen von ThinApp-Paketen zur Laufzeit auf Endbenutzercomputern erstellen. Durch diesen Prozess können Sie Komponententeile separat verpacken, implementieren und aktualisieren und dabei die Vorteile der Virtualisierung von Anwendungen erhalten.

ThinApp unterstützt die gleichzeitige Kombination von bis zu 250 Paketen. Jedes Paket kann von beliebiger Größe sein.

Sandbox-Änderungen von verknüpften Paketen sind für das Basispaket nicht sichtbar. Sie können zum Beispiel Acrobat Reader als eigenständiges virtuelles Paket installieren und als mit der Basisanwendung Firefox verknüpftes Paket. Wenn Sie Acrobat Reader als eigenständige Anwendung starten, indem Sie das virtuelle Paket ausführen und Änderungen in den Voreinstellungen vornehmen, speichert ThinApp die Änderungen in der Sandbox für Acrobat Reader. Wenn Sie Firefox starten, kann Firefox diese Änderungen nicht erkennen, weil Firefox eine eigene Sandbox hat. Beim Öffnen einer `PDF`-Datei mit Firefox werden die Änderungen der Voreinstellungen, die in der eigenständigen Anwendung Acrobat Reader vorhanden sind, nicht angezeigt.

Weitere Informationen über das Dienstprogramm Application Link erhalten Sie unter „Application Link-Updates“ auf Seite 43, „OptionalAppLinks“ auf Seite 91 und „RequiredAppLinks“ auf Seite 90.

Pfadnamenformate für Application Link

Das Dienstprogramm Application Link unterstützt die folgenden Pfadnamenformate:

- Pfadnamen können sich auf die ausführbare Basisdatei beziehen. Beispielsweise wird aus `RequiredAppLinks=..\SomeDirectory` der Pfadname `c:\MyDir\SomeDirectory`, wenn Sie die ausführbare Basisdatei unter dem Pfad `c:\MyDir\SubDir\ Dependency.exe` ausführen.
- Als Pfadnamen können absolute Pfadnamen verwendet werden. Beispiel: `RequiredAppLinks=c:\SomeDirectory`.
- Pfadnamen können eine Netzwerkfreigabe oder einen UNC-Pfad verwenden. Beispiel: `RequiredAppLinks=\\share\somedir\Dependency.exe`.
- Pfadnamen können Umgebungsvariablen enthalten und dynamisch auf alle vorstehenden Pfadnamen erweitert werden. Beispiel: `RequiredAppLinks=%MYAPP_ADDONS%\Dependency.exe`.
- Pfadnamen können mehrere Links oder Abhängigkeiten durch ein Semikolon angeben, mit dem einzelne Dateinamen voneinander getrennt werden. Beispiel: `RequiredAppLinks=Dependency1.exe; Dependency2.exe;`.

RequiredAppLinks

Der Parameter `RequiredAppLinks` gibt eine Liste mit erforderlichen Paketen an, die zur Laufzeit in das Basispaket importiert werden sollen. Sie können diesen Parameter in der `Package.ini`-Datei des Basispakets konfigurieren.

Schlägt der Importvorgang für ein abhängiges Paket fehl, so wird eine Fehlermeldung angezeigt und die ausführbare Basisdatei wird beendet. Sie können den Vorgang stattdessen mit dem Parameter `OptionalAppLinks` fortsetzen, selbst wenn Fehler beim Laden auftreten. Wenn Sie ein Paket mithilfe eines Platzhaltermusters angeben und Dateien nicht mit dem Platzhaltermuster übereinstimmen, generiert ThinApp keine Fehlermeldung.

Zum Importieren von Paketen müssen folgende Vorgänge ausgeführt werden:

- Ausführen aus VBScripts aus importierten Paketen
- Starten von Autostart-Diensten aus importierten Paketen
- Registrieren von Schriftarten aus importierten Paketen
- Verschieben der SxS DLL-Dateien von Windows XP zu Windows Vista

Sie müssen eine Verknüpfung zum primären Datencontainer eines Pakets erstellen. Sie können keine Verknüpfung zu anderen Verknüpfungspaketen erstellen.

Pfadnamen sind auf dem Arbeitscomputer, da die Verknüpfung sich bei Laufzeit auf den Clientcomputer auswirkt. Sie können absolute Pfade angeben, zum Beispiel `c:\abs\path\dotnet.exe`, relative Pfade, zum Beispiel `relpath\dotnet.exe`, und UNC-Pfade, zum Beispiel `\\server\share\dotnet.exe`. Pfadnamen können Umgebungsvariablen enthalten. Die verknüpften Pakete können mit einem Semikolon getrennt werden.

Weitere Informationen über das Dienstprogramm Application Link erhalten Sie unter „Application Link-Updates“ auf Seite 43.

Beispiele

Wenn Sie das .NET Framework im Paket `dotnet.exe` verpacken und eine .NET-Anwendung haben, können Sie angeben, dass die Anwendung mit der `dotnet.exe`-Datei verknüpft werden muss, bevor sie gestartet werden kann.

```
RequiredAppLinks=c:\abs\path\dotnet.exe
```

Sie können ein einzelnes Paket importieren, das in demselben Verzeichnis gespeichert ist wie die ausführbare Basisdatei.

```
RequiredAppLinks=Plugin.exe
```

Sie können ein einzelnes Paket importieren, das in demselben Unterverzeichnis gespeichert ist wie die ausführbare Basisdatei.

```
RequiredAppLinks=plugins\Plugin.exe
```

Sie können alle ausführbaren Dateien importieren, die in dem Verzeichnis für Plug-In-Dateien gespeichert sind. Wenn ThinApp eine ausführbare Datei nicht importieren kann, weil sich die Datei nicht in einem geeigneten ThinApp-Paket befindet oder weil ein Sicherheitsproblem aufgetreten ist, schlägt das Laden der ausführbaren Basisdatei fehl.

```
RequiredAppLinks=plugins\*.exe
```

Sie können alle ausführbaren Dateien importieren, die unter dem absoluten Pfad `n:\plugins` gespeichert sind.

```
RequiredAppLinks=n:\plugins\*.exe
```

Sie können die Umgebungsvariable `PLUGINS` erweitern und alle ausführbaren Dateien an diesem Speicherort importieren.

```
RequiredAppLinks=%PLUGINS%\*.exe
```

Sie können zwei angegebene Plug-In-Dateien laden und eine Liste mit ausführbaren Dateien, die am Speicherort für Plug-Ins gespeichert sind.

```
RequiredAppLinks=plugin1.exe;plugin2.exe;plugins\*.exe
```

OptionalAppLinks

Der Parameter `OptionalAppLinks` ist ähnlich wie der Parameter `RequiredAppLinks`, ignoriert jedoch Fehler und startet die Hauptanwendung auch dann, wenn ein Importvorgang fehlschlägt.

Sie müssen eine Verknüpfung zum primären Datencontainer eines Pakets erstellen. Sie können keine Verknüpfung zu anderen Verknüpfungspaketen erstellen.

Pfadnamen sind auf dem Arbeitscomputer, da die Verknüpfung sich bei Laufzeit auf den Clientcomputer auswirkt. Sie können absolute Pfade angeben, zum Beispiel `c:\abs\path\dotnet.exe`, relative Pfade, zum Beispiel `relpath\dotnet.exe`, und UNC-Pfade, zum Beispiel `\\server\share\dotnet.exe`.

Die Parameter `RequiredAppLinks` und `OptionalAppLinks` verwenden dieselbe Syntax. Weitere Informationen über den Parameter `RequiredAppLinks` und Beispiele erhalten Sie unter „RequiredAppLinks“ auf Seite 90.

Konfigurieren von Anwendungs-Updates mit Application Sync

Mit dem Dienstprogramm Application Sync können bereitgestellte virtuelle Anwendungen auf dem neuesten Stand gehalten werden. Wenn eine Anwendung gestartet wird, kann Application Sync einen Webserver abfragen, um zu ermitteln, ob eine aktualisierte Version des Pakets verfügbar ist. Sofern ein Update verfügbar ist, lädt ThinApp die Unterschiede zwischen dem vorhandenen Paket und dem neuen Paket herunter und erstellt eine aktualisierte Version des Pakets.

Das Dienstprogramm Application Sync lädt Updates im Hintergrund herunter. Sie können weiterhin eine alte Version der Anwendung benutzen. Beendet der Benutzer die Anwendung vor Abschluss des Downloadvorgangs, so wird der Downloadvorgang fortgesetzt, sobald die virtuelle Anwendung erneut gestartet wird. Im Anschluss an den Downloadvorgang aktiviert ThinApp die neue Version beim nächsten Starten der Anwendung.

Sie müssen den Parameter `AppSyncURL` kommentieren, um sämtliche Application Sync-Parameter zu aktivieren. Bei den folgenden Einträgen handelt es sich um Standardeinstellungen für Application Sync-Parameter:

```
AppSyncURL=https://example.com/some/path/PackageName.exe
AppSyncUpdateFrequency=1d
AppSyncExpirePeriod=30d
AppSyncWarningPeriod=5d
AppSyncWarningFrequency=1d
AppSyncWarningMessage=Diese Anwendung ist in AppSyncWarningPeriod Tagen nicht mehr für die
Benutzung verfügbar, wenn sie keine Verbindung zu ihrem Update-Server herstellen kann. Überprüfen
Sie Ihre Netzwerkverbindung, um den unterbrechungsfreien Dienst sicherzustellen
AppSyncExpireMessage=Diese Anwendung konnte AppSyncExpirePeriod Tage lang keine Verbindung zu
ihrem Update-Server herstellen und ist daher nicht mehr für die Benutzung verfügbar. Überprüfen
Sie Ihre Netzwerkverbindung und versuchen Sie es erneut
AppSyncUpdatedMessage=
AppSyncClearSandboxOnUpdate=0
```

Weitere Informationen über das Dienstprogramm Application Sync erhalten Sie unter „Application Sync-Updates“ auf Seite 41.

AppSyncClearSandboxOnUpdate

Der Parameter `AppSyncClearSandboxOnUpdate` leert die Sandbox im Anschluss an ein Update.

Beispiele

Der Standardwert für den Parameter `AppSyncClearSandboxOnUpdate` leert die Sandbox nicht.

```
AppSyncClearSandboxOnUpdate=0
```

Sie können die Sandbox nach dem Anwenden der Updates leeren.

```
AppSyncClearSandboxOnUpdate=1
```

AppSyncExpireMessage

Der Parameter `AppSyncExpireMessage` legt fest, welche Meldung angezeigt wird, wenn die Verbindung mit dem Webserver nach der Ablaufperiode fehlschlägt und eine virtuelle Anwendung gestartet wird. Die Anwendung wird beendet, wenn die Meldung angezeigt wird.

Beispiele

ThinApp liefert eine Standardmeldung für den Parameter `AppSyncExpireMessage`.

```
AppSyncExpireMessage=Diese Anwendung konnte <AnwSyncAblaufZeit_Wert> Tage lang keine Verbindung
zu ihrem Update-Server herstellen und ist daher nicht mehr für die Benutzung verfügbar.
Überprüfen Sie Ihre Netzwerkverbindung und versuchen Sie es erneut.
```

Ist der Wert des Parameters `AppSyncExpirePeriod` in Stunden oder Minuten angegeben, ändern Sie die Mitteilung, um Stunden oder Minuten anstatt Tage anzugeben.

AppSyncExpirePeriod

Der Parameter `AppSyncExpirePeriod` legt fest, nach wie vielen Minuten (m), Stunden (h), oder Tagen (d) das Paket abläuft. Kann ThinApp den Webserver für die Suche nach Updates nicht erreichen, so funktioniert das Paket so lange weiter, bis die Ablaufperiode beendet ist und der Benutzer es schließt. Selbst nachdem die Ablaufperiode endet, versucht ThinApp den Webserver nach jedem folgenden Startversuch zu erreichen.

Beispiele

Sie können verhindern, dass das Paket abläuft, indem Sie den Standardwert `never (nie)` angeben.

```
AppSyncExpirePeriod=never (nie)
```

AppSyncURL

Der Parameter `AppSyncURL` legt die URL des Webservers oder den Speicherort der Dateifreigaben fest, auf dem die aktualisierte Version einer Anwendung gespeichert ist. ThinApp überprüft diesen Speicherort und lädt das aktualisierte Paket herunter.

Application Sync funktioniert mit HTTP (unsicher), HTTPS (sicher) und Dateiprotokollen. Zum HTTPS-Protokoll gehört auch die Überprüfung der Identität des Webservers. Sie können einen Benutzernamen und ein Kennwort im Parameter `AppSyncURL` angeben, um eine Basisauthentifizierung zu gewährleisten. ThinApp übernimmt die Standardeinstellungen von Internet Explorer für den Proxyserver.

Sie müssen den Parameter `AppSyncURL` kommentieren, um sämtliche Application Sync-Parameter zu aktivieren.

Beispiele

Sie können dem Parameter `AppSyncURL` einen HTTP- oder HTTPS-Wert im folgenden Format zuweisen.

```
AppSyncURL=https://<Website.com>/<Pfad>/<package_name>.exe
```

Sie können Pfade zu lokalen Laufwerken und zu Netzlaufwerken angeben.

```
file:///C:/<Pfad>/<Paket_Name>.exe
```

Sie können einen UNC-Pfad angeben und auf die Speicherorte von Netzwerkressourcen zugreifen.

```
file://<Server>/<Freigabe>/<Pfad>/<Paket_Name>.exe
```

AppSyncUpdateFrequency

Der Parameter `AppSyncUpdateFrequency` gibt an, wie häufig ThinApp den Webserver auf Anwendungs-Updates überprüft. Sie können das Update-Intervall in Minuten (m), Stunden (h) oder Tagen (d) festlegen.

Ist dieselbe Sandbox für eine andere laufende Anwendung freigegeben, sucht ThinApp nicht nach einem Update.

Beispiele

Mit dem Standardwert wird ein Paket einmal täglich mit dem Webserver verbunden, um nach Updates zu suchen.

```
AppSyncUpdateFrequency=1d
```

Der Wert 0 legt fest, dass die gekapselte Anwendung bei jedem Starten der Anwendung nach Updates sucht.

```
AppSyncUpdateFrequency=0
```

AppSyncUpdatedMessage

Der Parameter `AppSyncUpdatedMessage` legt fest, welche Meldung beim ersten Starten eines aktualisierten Pakets angezeigt wird.

Beispiele

Der Wert AppSyncUpdatedMessage bestätigt, dass die Anwendung aktualisiert wurde.

```
AppSyncUpdatedMessage=Ihre Anwendung wurde aktualisiert.
```

AppSyncWarningFrequency

Der Parameter AppSyncWarningFrequency gibt an, wie häufig eine Warnmeldung angezeigt werden soll, bevor das Paket abläuft. Sie können den Zeitraum in Minuten (m), Stunden (h) oder Tagen (d) angeben.

Beispiele

Mit dem Standardwert wird festgelegt, dass die Warnmeldung nur einmal täglich angezeigt wird.

```
AppSyncWarningFrequency=1d
```

Der Wert 0 gibt an, dass die Warnmeldung bei jedem Starten der Anwendung angezeigt wird.

```
AppSyncWarningFrequency=0
```

AppSyncWarningMessage

Der Parameter AppSyncWarningMessage legt fest, welche Meldung angezeigt wird, wenn die Warnperiode beginnt. Beim ersten Starten der Anwendung in der Warnperiode wird eine Warnmeldung angezeigt und ThinApp versucht, über den Server auf die Updates zuzugreifen. Kann ThinApp das Paket nicht aktualisieren, so versucht ThinApp dies erneut bei jedem Starten der Anwendung. Die Warnmeldung wird nur nach dem Ablauf der einzelnen AppSyncWarningFrequency-Perioden angezeigt.

Beispiele

ThinApp enthält eine Standardmeldung für die Warnmeldung des Dienstprogramms Application Sync.

```
AppSyncWarningMessage=Diese Anwendung ist in %%remaining_days%% Tag(en) nicht mehr für die
Benutzung verfügbar, wenn sie keine Verbindung zu ihrem Update-Server herstellen kann. Überprüfen
Sie Ihre Netzwerkverbindung, um den unterbrechungsfreien Dienst sicherzustellen.
```

Die Variable %%remaining_days%% gibt die Anzahl der Tage an, die noch bis zum Ablauf des Pakets verbleiben.

Ist der Wert des Parameters AppSyncWarningPeriod in Stunden oder Minuten angegeben, ändern Sie die Mitteilung, um Stunden oder Minuten anstatt Tage anzugeben.

AppSyncWarningPeriod

Der Parameter AppSyncWarningPeriod legt den Beginn der Warnperiode fest, bevor das Pakets abläuft. Sie können den Zeitraum in Minuten (m), Stunden (h) oder Tagen (d) angeben. Wenn die Warnperiode beginnt, prüft ThinApp den Webserver jedes Mal wenn eine Anwendung gestartet wird und setzt den Wert des Parameters AppSyncUpdateFrequency auf 0.

Beispiele

Der Standardzeitraum des Parameters AppSyncWarningPeriod beträgt fünf Tage.

```
AppSyncWarningPeriod=5d
```

Konfigurieren von MSI-Dateien

MSI-Parameter konfigurieren MSI-Dateien, die Sie möglicherweise anstelle von ausführbaren Dateien installieren. Informationen über MSI-Dateien erhalten Sie unter „Erstellen einer MSI-Datenbank“ auf Seite 31.

MSIArpProductIcon

Der Parameter MSIArpProductIcon gibt an, welche Symbole in die Liste „Software“ der Windows-Systemsteuerung eingefügt werden sollen.

Beispiele

Sie können Symbole für Microsoft Office 2007 in der Liste „Software“ angeben.

```
MSIArpProductIcon=%Program Files Common%\Microsoft Shared\OFFICE12\
               Office Setup Controller\OSETUP.DLL,1
```

Das allgemeine Format lautet `MSIArpProductIcon=<Dateiname>[,<Symbol_Index>]`. Der Eintrag `<Symbol_Index>` ist optional.

MSIDefaultInstallAllUsers

Der Parameter `MSIDefaultInstallAllUsers` legt den Installationsmodus der MSI-Datenbank fest. Sie können eine `.msi`-Datei für alle Benutzer auf einem Computer und für einzelne Benutzer installieren. Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Datenbank für Windows Installer anfordert.

Informationen über die Erzwingung einer MSI-Installation für jeden Benutzer oder Rechner erhalten Sie unter „Erzwingen von MSI-Bereitstellungen für jeden Benutzer oder jeden Rechner“ auf Seite 33.

Beispiele

Installiert ein Benutzer die `.msi`-Datei mit dem Wert `1` für den Parameter `MSIDefaultInstallAllUsers`, so können der betreffende Benutzer und alle anderen auf dem Computer angemeldeten Benutzer Verknüpfungen, Dateitypzuordnungen usw. benutzen. Für die Installation eines Rechners müssen Sie über Administratorrechte verfügen.

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIDefaultInstallAllUsers=1
```

Installiert ein Benutzer die `.msi`-Datei mit dem Wert `0` für den Parameter `MSIDefaultInstallAllUsers`, so kann nur der betreffende Benutzer Verknüpfungen, Dateitypzuordnungen usw. benutzen. Für die Installation eines einzelnen Benutzers sind keine Administratorrechte erforderlich.

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIDefaultInstallAllUsers=0
```

Administratoren können eine Datenbankinstallation für alle Benutzer auf einem Rechner erstellen, und Benutzer ohne Administratorrechte können Installationen für einzelne Benutzer erstellen.

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIDefaultInstallAllUsers=2
```

MSIFilename

Der Parameter `MSIFilename` ermöglicht die Generierung einer MSI-Datenbank und gibt deren Dateinamen an.

Der Parameter `MSIFilename` erzeugt einen Windows Installer mit dem angegebenen Dateinamen im Ausgabeverzeichnis.

Beispiele

Sie können eine MSI-Datei während des Build-Prozesses generieren und die `mymsi.msi`-Datei durch Ihren eigenen Dateinamen ersetzen.

```
[BuildOptions]
MSIFilename=mymsi.msi
```

MSIInstallDirectory

Der Parameter `MSIInstallDirectory` gibt den Pfad des MSI-Installationsverzeichnisses an. Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Datenbank für Windows Installer anfordert.

Standardmäßig speichert ThinApp Pakete im Verzeichnis `%ProgramFilesDir%\<Bestandsname>` während der Installation auf jedem Rechner. Sie können den Installationspfad mit dem Parameter `MSIInstallDirectory` ändern. Verwenden Sie einen relativen Pfad, so ist der Pfad bei Installationen auf jedem Rechner relativ zu `%ProgramFilesDir%` und bei Installationen für jeden Benutzer ist er relativ zu `%AppData%`. Wenn Sie den Parameter `MSIInstallDirectory` auf `ExampleDir` festlegen, lautet das Standardinstallationsverzeichnis für Installationen auf jedem Rechner `%ProgramFilesDir%\ExampleDir`.

Beispiele

Sie können eine `.msi`-Datei im Verzeichnis `C:\Programme\Meine Anwendung` installieren.

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIInstallDirectory=Meine Anwendung
```

MSIManufacturer

Der Parameter `MSIManufacturer` gibt an, welcher Hersteller in die MSI-Datenbank eingegeben werden soll. Als Standardeinstellung wird der Name des Unternehmens angegeben, bei dem Ihre Windows-Lizenz registriert ist. Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Datenbank für Windows Installer anfordert.

Beispiele

Sie können für den Parameter `MSIManufacturer` den Namen Ihrer Organisation angeben. Der Name hat keine weiteren Auswirkungen, außer dass er in den Eigenschaften der MSI-Datenbank angezeigt wird.

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIManufacturer=Name meiner Firma
```

MSIProductCode

Der Parameter `MSIProductCode` gibt einen Produktcode für die MSI-Datenbank an. Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Datenbank für Windows Installer anfordert.

Für jede MSI-Datenbank ist ein Produktcode erforderlich. Beim Kapselungsprozess wird ein Standardproduktcode generiert und in die `Package.ini`-Datei eingefügt. Wenn Sie den Produktcode ändern, muss der neue Wert eine gültige global eindeutige Kennung (Globally Unique Identifier, GUID) sein.

Beispiele

Sie können eine `MSI`-Datei mit dem Produktcode 590810CE-65E6-3E0B-08EF-9CCF8AE20D0E erstellen.

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIProductCode={590810CE-65E6-3E0B-08EF-9CCF8AE20D0E}
```

MSIProductVersion

Der Parameter `MSIProductVersion` gibt eine Produktversionsnummer für die MSI-Datenbank an. Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Datenbank für Windows Installer anfordert.

Die Produktversion wird angezeigt, wenn Sie die Eigenschaften der Datenbank aufrufen. Wenn Sie ein Paket auf einem Rechner installieren, auf dem dasselbe Paket bereits installiert ist, überprüft Windows Installer die Versionsnummern und sperrt die Installation einer älteren Version gegenüber einer aktuelleren Version. Unter diesen Umständen müssen Sie die neue Version manuell deinstallieren.

Beispiele

Die Standardproduktversion lautet 1.0.

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIProductVersion=1.0
```

MSIRequireElevatedPrivileges

Der Parameter `MSIRequireElevatedPrivileges` gilt für Windows Vista und gibt die Anforderung erhöhter Rechte für die MSI-Datenbank an. Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Datenbank für Windows Installer anfordert.

Der Wert `1` kennzeichnet die MSI-Datenbank derart, dass diese erhöhte Rechte erfordert. Wenn Ihr System für UAC-Eingabeaufforderungen konfiguriert ist, wird eine UAC-Eingabeaufforderung angezeigt, wenn Sie eine Anwendung installieren.

Mit dem Wert `0` werden die UAC-Eingabeaufforderung und die Installation auf allen Rechnern gesperrt.

Beispiele

Mit der Standardeinstellung `1` wird eine MSI-Datei erstellt, die nach erhöhten Rechten in Windows Vista fragt.

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIRequireElevatedPrivileges=1
```

MSIUpgradeCode

Der Parameter `MSIUpgradeCode` gibt einen Upgrade-Code für die MSI-Datenbank an. Der Parameter funktioniert nur unter der Voraussetzung, dass der Parameter `MSIFilename` die Generierung einer Datenbank für Windows Installer anfordert.

VMware empfiehlt, jede einzelne MSI-Datenbank mit einem eigenen Upgrade-Code zu versehen. Beim Kapselungsprozess wird ein passender Upgrade-Code in der `Package.ini`-Datei generiert. Der Wert `UpgradeCode` sollte nicht geändert werden, es sei denn, Sie stellen fest, dass der neue Wert eine gültige GUID ist.

Beispiele

Sie können eine MSI-Datei mit dem Upgrade-Code D89F1994-A24B-3E11-0C94-7FD1E13AB93F erstellen.

```
[BuildOptions]
MSIFilename=mymsi.msi
MSIUpgradeCode={D89F1994-A24B-3E11-0C94-7FD1E13AB93F}
```

MSIUseCabs

Der Parameter `MSIUseCabs` legt die Verwendung von `.cab`-Dateien fest.

Wenn Sie den Wert `1` festlegen, speichert ThinApp die Paketdateien in einer `.cab`-Datei. Die `.cab`-Datei befindet sich in der MSI-Datei.

Wenn Sie den Wert `0` festlegen, verwendet ThinApp keine `.cab`-Dateien. Es kann sinnvoll sein, eine `.cab`-Datei zu vermeiden, wenn sich der Installationsprozess für Anwendungen hierdurch verlangsamen würde. Sie können die MSI-Datei und die einzelnen ausführbaren Dateien in `/bin` verteilen, um die Anwendung zu installieren.

Beispiele

Sie können Paketdateien in einer `.cab`-Datei speichern.

```
[BuildOptions]
MSIUseCabs=1
```

Konfigurieren von Sandbox-Speicher und Bestandsnamen

Die Sandbox-Parameter konfigurieren das Verzeichnis, in dem alle von der gekapselten Anwendung durchgeführten Änderungen gespeichert werden. Der ThinApp-Bestandsname kann bewirken, dass der Sandboxname geändert werden muss.

Weitere Informationen über die Sandbox erhalten Sie in Anhang B, „ThinApp-Sandbox“, auf Seite 101.

InventoryName

Der Parameter `InventoryName` ist eine Zeichenfolge, die Dienstprogramme für die Bestandsnachverfolgung zur Paketidentifizierung verwenden. Dieser Parameter bestimmt die Standardnamen des Projektordners und der Sandbox während der Kapselung der Anwendung.

Bei der Kapselung der Anwendung wird ein Standardwert für den Parameter `InventoryName` festgelegt. Dies erfolgt basierend auf neuen Zeichenfolgen, die unter einem der folgenden Speicherorte erstellt wurden:

- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`
- `HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall`

Das Dienstprogramm `thinreg.exe` und die MSI-Dateien von ThinApp referenzieren diesen Parameter, um den Produktnamen zu ermitteln, der unter „Software“ in der Systemsteuerung angezeigt werden soll. Lautet der Bestandsname beispielsweise `SuperApp` und installieren Sie eine MSI-Datei oder registrieren ein Paket mit dem Dienstprogramm `thinreg.exe`, so wird in der Liste „Software“ eine installierte Anwendung mit der Zeichenfolge `SuperApp (VMware ThinApp)` angezeigt. ThinApp hängt `VMware ThinApp` an den Bestandsnamen an, um Anwendungen zu unterscheiden, die bei Bestandsüberprüfungen virtualisiert wurden.

Sie können denselben Bestandsnamen über verschiedene Versionen derselben Anwendung hinweg verwenden, um sicherzustellen, dass nur die neueste Version in der Liste „Software“ angezeigt wird. Die Anwendungen in der Liste „Software“ überschreiben einander und verhindern, dass alle registrierten Pakete deinstalliert werden. Möchten Sie mehr als eine Version deinstallieren, müssen Sie für jede Version einen eigenen Bestandsnamen verwenden. Verwenden Sie beispielsweise Microsoft Office 2003 und Microsoft Office 2007 als Bestandsnamen anstatt nur Microsoft Office. Wenn Sie verschiedene Versionen einer virtuellen Anwendung in derselben Umgebung pflegen, sollten Sie möglicherweise den Parameter `SandboxName` ändern, um sicherzustellen, dass eine neue Version die Benutzereinstellungen in einer anderen Sandbox isoliert hat.

Wenn Sie ein Paket haben, das andere Anwendungen einschließt, müssen Sie möglicherweise den Bestandsnamen manuell aktualisieren, damit der wirkliche Inhalt des Pakets wiedergegeben wird. Wenn Sie beispielsweise die Anwendung `SuperApp` kapseln und das Paket Java Runtime einschließt, kann als Wert für `InventoryName` `Java Runtime Environment 1.5` anstelle von `SuperApp` angezeigt werden. Die Liste „Software“ zeigt die erste im Paket installierte Anwendung an.

Beispiele

Sie können den Bestandsnamen „Microsoft Office 2003“ festlegen.

```
[BuildOptions]
InventoryName=Microsoft Office 2003
```

RemoveSandboxOnExit

Der Parameter `RemoveSandboxOnExit` löscht die Sandbox und setzt die Anwendung zurück, wenn der letzte untergeordnete Prozess beendet wird.

ThinApp speichert alle Änderungen der Anwendung in der Registrierung und an den Speicherorten des Dateisystems mit dem Isolationsmodus „WriteCopy“ oder „Full (Voll)“ in der Sandbox. Standardmäßig behält das Sandbox-Verzeichnis die Einstellungen über mehrere Ausführungen der Anwendung einheitlich bei. Möglicherweise möchten Sie die Sandbox bei jedem Beenden der Anwendung löschen.

Wenn die Anwendung untergeordnete Prozesse erstellt, löscht ThinApp die Sandbox erst, nachdem alle untergeordneten Prozesse beendet wurden. Anwendungen können so ausgelegt sein, dass die untergeordneten Prozesse bestehen bleiben und die Bereinigung darum blockieren. Zum Beispiel behält Microsoft Office 2003 den Prozess `ctfmon.exe` bei. Eventuell müssen Sie den Prozess `ctfmon.exe` und die untergeordneten Prozesse mithilfe eines Skripts beenden, um die Ausführung der Bereinigung zu erzwingen.

Sie können zur Laufzeit entscheiden, ob die API-Funktion des Skripts `RemoveSandboxOnExit` zum Löschen der Sandbox bei Beenden verwendet werden soll.

Beispiele

Sie können die Sandbox löschen, wenn die Anwendung beendet wird.

```
[BuildOptions]
RemoveSandboxOnExit=1
```

Sie können die Sandbox beibehalten, wenn die Anwendung beendet wird. Dies ist das Standardverhalten.

```
[BuildOptions]
RemoveSandboxOnExit=0
```

SandboxName

Der Parameter `SandboxName` legt den Namen des Verzeichnisses fest, in dem die Sandbox gespeichert wird.

Beim Upgrade einer Anwendung können Sie mithilfe des Sandbox-Namens ermitteln, ob Benutzer die vorherigen persönlichen Einstellungen beibehalten oder neue Einstellungen benötigen. Änderungen des Sandbox-Namens bei neuen Installationen wirken sich auf die Notwendigkeit zur Erstellung einer neuen Sandbox mit anderen Einstellungen oder zur Beibehaltung derselben Sandbox aus.

Beispiele

Sie können als Sandbox-Verzeichnisnamen `Meine Anwendung 1.0` angeben.

```
[BuildOptions]
SandboxName=Meine Anwendung 1.0
```

SandboxNetworkDrives

Der Parameter `SandboxNetworkDrives` ermittelt, ob ThinApp Sandboxen für zugeordnete Netzlaufwerke verwendet.

Beispiele

Sie können Änderungen in der Sandbox speichern und verhindern, dass die Benutzer direkt auf zugeordnete Netzlaufwerke schreiben.

```
[BuildOptions]
SandboxNetworkDrives=1
```

Sie können direkt auf zugeordnete Netzlaufwerke schreiben, ohne die Änderungen in einer Sandbox zu speichern. Dies ist das Standardverhalten.

```
(Standard)
[BuildOptions]
SandboxNetworkDrives=0
```

SandboxPath

Der Parameter `SandboxPath` legt den Pfad fest, unter dem eine neue Sandbox erstellt wird.

Wenn eine Anwendung nur von einem portablen Speichermedium aus ausgeführt wird, zum Beispiel von einem USB-Flash-Gerät, können Sie mithilfe des Parameters `SandboxPath` erzwingen, dass die Anwendung eine lokale Sandbox verwendet. Informationen darüber, wie ThinApp eine Sandbox findet, erhalten Sie unter „Suchreihenfolge für die Sandbox“ auf Seite 101.

Beispiele

Sie können die Sandbox in demselben Verzeichnis wie die ausführbare Datei erstellen.

```
[BuildOptions]
SandboxPath=.
```

Sie können die Sandbox in einem Unterverzeichnis erstellen, das dem Speicherort der ausführbaren Datei untergeordnet ist.

```
[BuildOptions]
SandboxPath=LocalSandbox\Subdir1
```

Sie können die Sandbox im Ordner `AppData` des Benutzers unter dem Unterverzeichnis `Thinstall` erstellen:

```
[BuildOptions]
SandboxPath=%AppData%\Thinstall
```

Sie können die Sandbox auf einem Laufwerk mit Netzwerkmapping erstellen.

```
[BuildOptions]
SandboxPath=Z:\Sandboxes
```

SandboxRemovableDisk

Der Parameter `SandboxRemovableDisk` bestimmt, ob ThinApp Änderungen der Wechseldatenträger auf die Datenträger oder in die Sandbox speichern kann. Wechseldatenträger sind zum Beispiel USB-Flash-Geräte und externe Festplatten.

Beispiele

Der Standardwert weist ThinApp an, Änderungen der Wechseldatenträgerdatei direkt auf den Datenträger zu schreiben.

```
[BuildOptions]
SandboxRemovableDisk=0
```

ThinApp kann Isolationsmodi auf Wechseldatenträger anwenden. Abhängig vom Isolationsmodus können sich Änderungen an Dateien, die auf Wechseldatenträgern gespeichert sind, in der Sandbox oder auf dem Wechseldatenträger befinden.

```
[BuildOptions]
SandboxRemovableDisk=1
```

ThinApp-Sandbox

B

Die Sandbox ist das Verzeichnis, in dem alle von der gekapselten Anwendung durchgeführten Änderungen gespeichert werden. Sobald Sie die Anwendung das nächste Mal starten, werden diese Änderungen aus der Sandbox integriert. Wenn Sie das Sandbox-Verzeichnis löschen, wird die Anwendung auf den gekapselten Status zurückgesetzt.

Dieser Abschnitt umfasst die folgenden Themen:

- „Suchreihenfolge für die Sandbox“ auf Seite 101
- „Steuern des Sandbox-Speicherorts“ auf Seite 103
- „Sandbox-Struktur“ auf Seite 104

Suchreihenfolge für die Sandbox

Beim Starten der gekapselten Anwendung sucht ThinApp an bestimmten Speicherorten und in einer bestimmten Reihenfolge nach einer bereits vorhandenen Sandbox. ThinApp verwendet die Sandbox, die als erste erkannt wird. Kann ThinApp keine vorhandene Sandbox ermitteln, erstellt ThinApp eine Sandbox gemäß bestimmten Umgebungsvariablen und Parametereinstellungen. Überprüfen Sie die Suchreihenfolge und Erstellungslogik der Sandbox, bevor Sie die Platzierung der Sandbox verändern.

In dieser Suchreihenfolge wird Mozilla Firefox 3.0 als Beispiel mit folgenden Variablen verwendet:

- `<Sandbox_Name>` ist Mozilla Firefox 3.0

  Der `SandboxName`-Parameter in der `Package.ini`-Datei legt den Namen fest. Siehe „SandboxName“ auf Seite 99.

- `<Sandbox_Pfad>` ist `Z:\sandboxes`

  Der `SandboxPath`-Parameter in der `Package.ini`-Datei legt den Pfad fest. Siehe „SandboxPath“ auf Seite 100.

- `<exe_Verzeichnis>` ist `C:\Programme\Firefox`

  Die Anwendung startet von diesem Speicherort aus.

- `<Computer_Name>` ist MAXMUSTERMANN-COMPUTER

- `%AppData%` ist `C:\Dokumente und Einstellungen\MaxMustermann\Anwendungsdaten`

  ThinApp fordert vom Betriebssystem den Ordnerspeicherort `Anwendungsdaten`. Der Speicherort hängt vom Betriebssystem oder der Konfiguration ab.

ThinApp startet die Sandbox-Suche, indem in dieser Reihenfolge nach folgenden Umgebungsvariablen gesucht wird:

- %<Sandbox_Name>_SANDBOX_DIR%

  Diese Umgebungsvariable ändert den Speicherort der Sandbox für bestimmte Anwendungen auf dem Computer. Ist zum Beispiel die Umgebungsvariable `Mozilla Firefox 3.0_SANDBOX_DIR` vorhanden, bestimmt ihr Wert den Speicherort des übergeordneten Verzeichnisses der Sandbox. Ist der Wert `z:\FirefoxSandbox`, bevor Sie die Anwendung starten, speichert ThinApp die Sandbox in `z:\FirefoxSandbox.MAXMUSTERMANN-COMPUTER`, wenn das Verzeichnis bereits besteht. Ist das Verzeichnis nicht vorhanden, erstellt ThinApp eine Sandbox in `z:\FirefoxSandbox`.

- %THINSTALL_SANDBOX_DIR%

  Diese Umgebungsvariable ändert den Speicherort jeder Sandbox auf einem Computer. Ist zum Beispiel die Umgebungsvariable `THINSTALL_SANDBOX_DIR` vorhanden, bestimmt ihr Wert den Speicherort des übergeordneten Verzeichnisses der Sandbox. Ist der Wert `z:\MySandboxes`, bevor Sie die Anwendung starten, erstellt ThinApp eine Sandbox in `z:\MySandboxes`.

Wenn ThinApp die Umgebungsvariable %<Sandbox_Name>_SANDBOX_DIR% oder %THINSTALL_SANDBOX_DIR% nicht erkennt, überprüft ThinApp die folgenden Dateisystemverzeichnisse und erstellt eine Sandbox im Verzeichnis, das als erstes erkannt wird:

- <exe_Verzeichnis>\<Sandbox_Name>.<Computer_Name>

  Zum Beispiel `C:\Programme\Firefox\Mozilla Firefox 3.0.MAXMUSTERMANN-COMPUTER`

- <exe_Verzeichnis>\<Sandbox_Name>

  Zum Beispiel `C:\Programme\Firefox\Mozilla Firefox 3.0`

- <exe_Verzeichnis>\Thinstall\<Sandbox_Name>.<Computer_Name>

  Zum Beispiel `C:\Programme\Firefox\Thinstall\Mozilla Firefox 3.0.MAXMUSTERMANN-COMPUTER`

- <exe_Verzeichnis>\Thinstall\<Sandbox_Name>

  Zum Beispiel `C:\Programme\Firefox\Thinstall\Mozilla Firefox 3.0`

- <Sandbox_Pfad>\<Sandbox_Name>.<Computer_Name>

  Zum Beispiel `Z:\sandboxes\Mozilla Firefox 3.0.MAXMUSTERMANN-COMPUTER`

- <Sandbox_Pfad>\<Sandbox_Name>

  Zum Beispiel `Z:\sandboxes\Mozilla Firefox 3.0`

- %AppData%\Thinstall\<Sandbox_Name>.<Computer_Name>

  Zum Beispiel `C:\Dokumente und Einstellungen\MaxMustermann\Anwendungsdaten\Thinstall\Mozilla Firefox 3.0.MAXMUSTERMANN-COMPUTER`

- %AppData%\Thinstall\<Sandbox_Name>

  Zum Beispiel `C:\Dokumente und Einstellungen\MaxMustermann\Anwendungsdaten\Thinstall\Mozilla Firefox 3.0`

Wenn ThinApp die Umgebungsvariable %<Sandbox_Name>_SANDBOX_DIR% oder %THINSTALL_SANDBOX_DIR% und die angegebenen Dateisystemverzeichnisse nicht erkennt, erstellt ThinApp eine Sandbox mit folgenden Richtlinien in dieser Reihenfolge:

- Ist der Parameter `SANDBOXPATH` `Package.ini` festgelegt, bestimmt dieser Wert den Sandbox-Speicherort.
- Schließt ThinApp die Sandbox-Suche ergebnislos ab, erstellt ThinApp eine Sandbox im Standardverzeichnis `%AppData%\Thinstall` des Benutzers.

ANMERKUNG Es kann nur jeweils ein Computer eine freigegebene Sandbox verwenden. Verwendet ein Computer bereits eine Sandbox, erstellt ThinApp eine neue Sandbox, damit Sie mit Ihrer Arbeit fortfahren können, bis die vorherige Kopie der Sandbox geschlossen wird.

Steuern des Sandbox-Speicherorts

Der Setup Capture-Prozess fügt den `SandboxName`-Parameter zur `Package.ini`-Datei hinzu. Wenn Sie Firefox kapseln und `Mozilla Firefox 3.0` der Wert dieses Parameters ist, lautet der Standardspeicherort der Sandbox für die Anwendung `%AppData%\Thinstall\Mozilla Firefox 3.0`. Der typische `%AppData%`-Speicherort ist `C:\Dokumente und Einstellungen\<Benutzer_Name>\Anwendungsdaten`. `%AppData%` wird oft einem freigegebenen Netzlaufwerk zugeordnet.

Speichern der Sandbox im Netzwerk

Sie können den `SandboxPath`-Parameter verwenden, um die Sandbox auf einem zugeordneten Laufwerk zu speichern. Ein Netzwerkspeicherort ist nützlich, um die Sandbox zu sichern und ebenso für Benutzer, die sich an einem beliebigen Computer anmelden und ihre Anwendungseinstellungen beibehalten möchten. Weitere Informationen über den `SandboxPath`-Parameter finden Sie unter „SandboxPath“ auf Seite 100.

So speichern Sie die Sandbox auf einem zugeordneten Laufwerk

1 Öffnen Sie die `Package.ini`-Datei.

2 Legen Sie unter dem `SandboxName`-Parameter den `SandboxPath`-Parameter auf den Netzwerkspeicherort fest.

```
SandboxName=Mozilla Firefox 3.0
SandboxPath=Z:\Sandbox
```

Ist zum Beispiel `Mozilla Firefox 3.0` der Wert des `SandboxName`-Parameters, erstellt die gekapselte Firefox-Anwendung die Sandbox unter `Z:\Sandbox\Mozilla Firefox 3.0`.

Speichern der Sandbox auf einem portablen Gerät

Sie können den `SandboxPath`-Parameter verwenden, um einen Sandbox-Speicherort auf einem portablen Gerät festzulegen. Sie können jedes portable Gerät wie zum Beispiel ein USB-Laufwerk verwenden, das als Festplattenlaufwerk im Systemordner `Arbeitsplatz` angezeigt wird. Ein Speicherort auf einem portablen Gerät ist nützlich, um die Sandbox-Daten auf dem Gerät zu speichern, auf dem sich die Anwendung befindet.

Weitere Informationen über den `SandboxPath`-Parameter finden Sie unter „SandboxPath“ auf Seite 100.

So speichern Sie die Sandbox auf einem USB-Laufwerk im selben Verzeichnis wie die ausführbare Datei

1 Öffnen Sie die `Package.ini`-Datei.

2 Legen Sie unter dem `SandboxName`-Parameter den `SandboxPath`-Parameter auf diesen Wert fest.

```
SandboxName=Mozilla Firefox 3.0
SandboxPath=.
```

Ist zum Beispiel `Mozilla Firefox 3.0` der Wert des `SandboxName`-Parameters, erstellt die gekapselte Firefox-Anwendung die Sandbox `Mozilla Firefox 3.0` im gleichen Verzeichnis, von dem aus Firefox ausgeführt wird.

So speichern Sie die Sandbox in einem Thinstall-Verzeichnis auf einem USB-Laufwerk auf derselben Ebene wie die ausführbare Datei

1 Ist die Umgebungsvariable `%THINSTALL_SANDBOX_DIR%` oder `%<Sandbox_Name>_SANDBOX_DIR%` festgelegt, deaktivieren Sie die Variablen.

2 Erstellen Sie ein `Thinstall`-Verzeichnis auf dem portablen Gerät im selben Verzeichnis wie Ihre gekapselte Anwendung.

Wenn die paketierte Anwendung das nächste Mal von einem portablen Gerät aus gestartet wird, erstellt sie eine Sandbox im `Thinstall`-Verzeichnis.

3 Wurden die Anwendung und die Sandbox ursprünglich von einem anderen Speicherort, zum Beispiel einem Computer, gestartet und Sie benötigen dieselbe Sandbox auf einem portablen Gerät, kopieren Sie das `Thinstall`-Verzeichnis von `%AppData%` auf das Verzeichnis, in dem sich die ausführbare Datei auf dem Gerät befindet.

ThinApp verwendet die Sandbox am ursprünglichen Speicherort nicht mehr.

Sandbox-Struktur

ThinApp verwendet zum Speichern der Sandbox fast dieselbe Dateistruktur wie die Struktur zum Erstellen des Projekts. ThinApp verwendet Makronamen für Speicherorte von Shell-Ordnern wie beispielsweise `%AppData%`, anstatt hartcodierter Pfade. Mithilfe dieser Struktur kann die Sandbox dynamisch in verschiedene Computer migriert werden, wenn die Anwendung von neuen Speicherorten gestartet wird.

Die Sandbox enthält die folgenden Registrierungsdateien:

- `Registry.rw.tvr` – Enthält alle Registrierungsänderungen, die die Anwendung vornimmt.
- `Registry.rw.lck` – Verhindert, dass andere Computer gleichzeitig eine auf einer Netzwerkfreigabe gespeicherte Registrierung verwenden.
- `Registry.tvr.backup` – Enthält eine Sicherungsdatei der `.tvr`-Datei, die ThinApp verwendet, wenn die ursprüngliche `.tvr`-Datei beschädigt ist.

Neben diesen Registrierungsdateien enthält die Sandbox Verzeichnisse, zu denen `%AppData%`, `%ProgramFilesDir%` und `%SystemRoot%` gehören. Jeder dieser Ordner enthält Änderungen der entsprechenden Ordner in der gekapselten Anwendung.

Änderungen an der Sandbox

ThinApp speichert Dateisysteminformationen in der virtuellen Registrierung. Die virtuelle Registrierung ermöglicht es ThinApp, den Dateisystemzugriff in der virtuellen Umgebung zu optimieren. Versucht zum Beispiel eine Anwendung eine Datei zu öffnen, muss ThinApp nicht im realen Dateisystem nach dem realen Speicherort und noch einmal nach dem Speicherort der Sandbox suchen. Stattdessen kann ThinApp alleine anhand der virtuellen Registrierung prüfen, ob die Datei vorhanden ist. Dadurch erhöht sich die Laufzeitleistung von ThinApp.

VMware bietet keine Unterstützung für das direkte Ändern oder Hinzufügen von Dateien zur Sandbox. Wenn Sie Dateien in das Sandbox-Verzeichnis kopieren, sind die Dateien für die Anwendung nicht sichtbar. Ist die Datei bereits in der Sandbox vorhanden, können Sie sie überschreiben und aktualisieren. VMware empfiehlt, dass Sie sämtliche Änderungen aus der Anwendung selbst vornehmen.

Auflisten virtueller Registrierungsinhalte mit vregtool

Da die Sandbox die Änderungen der Registrierung enthält, benötigen Sie möglicherweise das Dienstprogramm `vregtool`, um virtuelle Registrierungsänderungen anzuzeigen. Sie müssen über Zugriff auf das Dienstprogramm `vregtool` unter `C:\Programme\VMware\VMware ThinApp` verfügen.

Ein Beispielbefehl zur Auflistung der Inhalte einer virtuellen Registrierungsdatei ist `vregtool registry.rw.tvr printkeys`.

Snapshot-Befehle und -Anpassungen C

Das Dienstprogramm `snapshot.exe` erstellt einen Snapshot eines Computerdateisystems und einer Registrierung und erstellt von zwei zuvor gekapselten Snapshots ein ThinApp-Projekt. Sie müssen das Dienstprogramm `snapshot.exe` nicht direkt starten, da der Setup Capture-Assistent es startet. Nur fortgeschrittene Benutzer und Systemintegratoren, die ThinAppOnly-Funktionalität in andere Plattformen integrieren, sollten dieses Dienstprogramm direkt verwenden.

Das Erstellen eines Snapshots von einem Computerdateisystem und einer Registrierung umfasst das Überprüfen und Speichern einer Kopie der folgenden Daten:

- Dateiinformationen für alle lokalen Laufwerke

  Diese Informationen umfassen Verzeichnisse, Dateinamen, Dateiattribute, Dateigrößen und Dateiänderungsdaten.

- Die Registrierungsstrukturen `HKEY_LOCAL_MACHINE` und `HKEY_USERS`

  ThinApp überprüft nicht die Registrierungseinträge `HKEY_CLASSES_ROOT` und `HKEY_CURRENT_USER` , weil diese Einträge Untereinträge der Einträge `HKEY_LOCAL_MACHINE` und `HKEY_USERS` sind.

Die `snapshot.ini`-Konfigurationsdatei gibt an, welche Verzeichnisse und Unterschlüssel von einem ThinApp-Projekt auszuschließen sind, wenn eine Anwendung gekapselt wird. Sie können diese Datei für bestimmte Anwendungen anpassen.

Dieser Abschnitt umfasst die folgenden Themen:

- „Methoden zur Verwendung des Dienstprogramms snapshot.exe“ auf Seite 105
- „Beispiele für snapshot.exe-Befehle“ auf Seite 107
- „Erstellen eines Projekts ohne den Setup Capture-Assistenten“ auf Seite 107
- „Anpassen der Snapshot.ini-Datei“ auf Seite 108

Methoden zur Verwendung des Dienstprogramms snapshot.exe

Mit dem Dienstprogramm `snapshot.exe` können Sie Snapshot-Dateien von Computerstatus, die Vorlagendatei für die `Package.ini`-Datei oder ein ThinApp-Projekt erstellen und den Inhalt einer Snapshot-Datei anzeigen.

Informationen über den vollständigen Vorgang zum Erstellen eines ThinApp-Projekts von der Befehlszeile finden Sie unter „Erstellen eines Projekts ohne den Setup Capture-Assistenten“ auf Seite 107.

Erstellen von Snapshots des Computerstatus

Das Dienstprogramm `snapshot.exe` erstellt eine Snapshot-Datei des Computerzustandes. ThinApp kapselt den Computerzustand und speichert ihn in einer einzelnen Datei, um ein Projekt zu erstellen. Das Dienstprogramm `snapshot.exe` speichert eine Kopie von Registrierungsdaten und Dateisystem-Metadaten, die Pfade, Dateinamen, Größen, Attribute und Zeitstempel umfassen.

Verwendung

```
snapshot.exe SnapshotFileName.snapshot [–Config ConfigFile.ini][BaseDir1][BaseDir2][BaseReg1]
```

Beispiele

```
Snapshot My.snapshot
Snapshot My.snapshot –Config MyExclusions.ini
Snapshot My.snapshot c:\MyAppDirectory HKEY_LOCAL_MACHINE\Software\MyApp
```

Optionen

Die Optionen geben die Verzeichnisse oder Unterschlüssel im Snapshot an.

Option	Beschreibung
–Config ConfigFile.ini	Gibt Verzeichnisse oder Registrierungsunterschlüssel an, die bei der Erstellung des Snapshots ausgeschlossen werden sollen. Wenn Sie keine Konfigurationsdatei angeben, verwendet ThinApp die `snapshot.ini`-Datei aus dem ThinApp-Installationsverzeichnis.
BaseDir1	Gibt eines oder mehrere Basisverzeichnisse an, die in die Überprüfung aufgenommen werden sollen. Wenn Sie keine Basisverzeichnisse angeben, überprüft das Dienstprogramm `snapshot.exe c:\` und alle Unterverzeichnisse. Wenn Sie einen Computer überprüfen, auf dem Windows oder Programmdateien auf verschiedenen Festplatten installiert sind, schließen Sie diese Laufwerke in Ihre Überprüfung ein. Wenn Sie wissen, dass Ihre Anwendungsinstallation Dateien an festen Speicherorten erstellt oder ändert, geben Sie diese Verzeichnisse an, um die Gesamtzeitdauer für das Überprüfen des Computers zu verringern.
BaseReg1	Gibt einen oder mehrere Registrierungsunterschlüssel an, die in die Überprüfung aufgenommen werden sollen. Wenn Sie keine Registrierungsunterschlüssel angeben, überprüft das Dienstprogramm `snapshot.exe` die Schlüssel `HKEY_LOCAL_MACHINE` und `HKEY_USERS`.

Erstellen der Template Package.ini-Datei aus zwei Snapshot-Dateien

Das Dienstprogramm `snapshot.exe` generiert eine `Package.ini`-Vorlagendatei. Das Dienstprogramm überprüft die zwei Snapshot-Dateien auf alle Anwendungen, die erstellt und auf die von Verknüpfungslinks oder dem Menü **Start** verwiesen wird. Die `Package.ini`-Vorlagendatei wird in einem ThinApp-Projekt zur Basis für die `Package.ini`-Datei.

Verwendung

```
snapshot.exe Snap1.snapshot –SuggestProject Snap2.snapshot OutputTemplate.ini
```

Beispiele

```
Snapshot Start.snapshot –SuggestProject End.snapshot Template.ini
```

ThinApp erfordert sämtliche Parameter.

Erstellen des ThinApp-Projekts aus der Template Package.ini-Datei

Das Dienstprogramm `snapshot.exe` erstellt die ThinApp-Projektdatei von der `Package.ini`-Vorlagendatei.

Verwendung

```
snapshot.exe Template.ini –GenerateProject OutDir [–Config ConfigFile.ini]
```

Beispiele

```
Snapshot Template.ini –GenerateProject c:\MyProject
               Snapshot Template.ini –GenerateProject c:\MyProject –Config MyExclusions.ini
```

`–Config ConfigFile.ini` ist optional. Die Konfigurationsdatei gibt Verzeichnisse oder Registrierungsunterschlüssel an, die vom Projekt ausgeschlossen werden sollen. Wenn Sie keine Konfigurationsdatei angeben, verwendet ThinApp die `snapshot.ini`-Datei.

Anzeigen von Inhalten einer Snapshot-Datei

Das Dienstprogramm `snapshot.exe` listet die Inhalte der Snapshot-Datei.

Verwendung

```
snapshot.exe SnapshotFileName.snapshot –Print
```

Beispiele

```
Snapshot Start.snapshot –Print
```

ThinApp erfordert sämtliche Parameter.

Beispiele für snapshot.exe-Befehle

Tabelle C-1 beschreibt Beispielbefehle für das Dienstprogramm `snapshot.exe`. Die Parameter unterscheiden nicht zwischen Groß- und Kleinschreibung. Die Befehle sind aus Platzgründen in der Befehlsspalte eingepackt.

Tabelle C-1. snapshot.exe-Beispielbefehle

Befehl	Beschreibung
`snapshot c:\Capture.snapshot`	Kapselt einen vollständigen Snapshot der lokalen Laufwerke und der Registrierung in der Datei `c:\Capture.snapshot`.
`snapshot c:\Capture.snapshot c:\ e:\`	Kapselt einen vollständigen Snapshot der Laufwerke `c:\` und `e:\`. ThinApp kapselt keine Registrierungsdaten.
`snapshot c:\Capture.snapshot c:\ HKEY_LOCAL_MACHINE\Software\Classes`	Kapselt einen vollständigen Snapshot des Laufwerkes `c:\` und der gesamten `HKEY_CLASSES_ROOT`-Registrierungsunterstruktur.
`snapshot c:\Original.snapshot –Diff c:\NewEnvironment.snapshot c:\MyProject`	Generiert ein ThinApp-Projektverzeichnis mithilfe des Vergleichs zwischen zwei Snapshots.
`snapshot Original.snapshot –DiffPrint NewEnvironment.snapshot`	Zeigt die Unterschiede zwischen zwei gekapselten Snapshots an.
`snapshot C:\data.snapshot snapshot C:\data.snapshot C:\ HKEY_LOCAL_MACHINE`	Speichert den Zustand des Computer-Dateisytems und der Registrierung.
`snapshot C:\start.snapshot –diffprint C:\end.snapshot`	Vergleicht zwei gespeicherte Zustände.
`snapshot C:\start.snapshot –print`	Druckt die Inhalte eines gespeicherten Zustandes.
`snapshot C:\start.snapshot –SuggestProject C:\end.snapshot C:\project.ini snapshot C:\project.ini –GenerateProject`	Generiert ein ThinApp-Projekt mithilfe des Vergleichs zwischen zwei gespeicherten Zuständen.

Erstellen eines Projekts ohne den Setup Capture-Assistenten

Sie können das Dienstprogramm `snapshot.exe` aus der Befehlszeile verwenden, statt des Setup Capture-Assistenten, der das Dienstprogramm `snapshot.exe` im Hintergrund ausführt. Das Befehlszeilen-Dienstprogramm ist nützlich, um eine große Anzahl an Anwendungen zu verpacken oder die ThinApp-Projekterstellung zu automatisieren. Der typische Speicherort für das Dienstprogramm `snapshot.exe` lautet `C:\Programme\VMware\VMware ThinApp\snapshot.exe`.

Der Snapshot-Prozess erstellt eine Kopie aller Registrierungseinträge auf dem System und in den Dateisystem-Metadaten. Zu den Dateisystem-Metadaten gehören Pfad, Dateiname, Attribut, Größe und Zeitstempel; tatsächliche Dateidaten werden jedoch ausgeschlossen.

So erstellen Sie ein Projekt mit dem Befehlszeilen-Dienstprogramm snapshot.exe

1 Speichern Sie einen Snapshot der aktuellen Computerkonfiguration auf der Festplatte.

```
snapshot.exe c:\Start.snapshot
```

2 Installieren Sie die Anwendung und nehmen Sie die erforderlichen manuellen Systemänderungen vor.

3 Speichern Sie einen Snapshot der neuen Computerkonfiguration auf der Festplatte.

```
snapshot.exe c:\End.snapshot
```

4 Generieren Sie eine `Package.ini`-Vorlagendatei.

```
snapshot.exe c:Start.snapshot –SuggestProject c:End.snapshot c:emplate.ini
```

ThinApp verwendet die Vorlagendatei, um die endgültige `Package.ini`-Datei zu generieren. Die Vorlagendatei enthält eine Liste von allen erkannten ausführbaren Dateieinstiegspunkte und `Package.ini`-Parameter. Wenn Sie Ihr eigenes Script schreiben und den Setup Capture-Assistenten ersetzen, verwenden Sie die `Package.ini`-Vorlagendatei, um die Einstiegspunkte auszuwählen, und die `Package.ini`-Parameter wie `InventoryName` zu erhalten oder anzupassen.

5 Generieren eines ThinApp-Projekts.

```
snapshot.exe c:\Template.ini –GenerateProject c:\MyProjectDirectory
```

6 (Optional) Löschen Sie die temporären Dateien `c:\Start.snapshot`, `c:\End.snapshot` und `c:\Template.ini`.

7 (Optional) Um mehrere Projekte mit unterschiedlichen Konfigurationen zu generieren, verwenden Sie die ursprüngliche `Start.snapshot`-Datei erneut und wiederholen Sie den Vorgang von Schritt 2.

Anpassen der Snapshot.ini-Datei

Die `snapshot.ini`-Konfigurationsdatei gibt an, welche Registrierungsschlüssel von einem ThinApp-Projekt auszuschließen sind, wenn eine Anwendung gekapselt wird.

Wenn Sie zum Beispiel Internet Explorer 7 verwenden, müssen Sie ThinApp möglicherweise veranlassen, folgende Registrierungsschlüssel zu kapseln:

- `HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Desktop\Components`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections`
- `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Hardware Profiles\0001\Software\Microsoft\windows\CurrentVersion\Internet Settings`

Wenn die `snapshot.ini`-Datei den `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections`-Schlüssel standardgemäß ausschließt, können Sie diesen Schlüssel aus der `snapshot.ini`-Datei entfernen, um zu gewährleisten, dass ThinApp den Schlüssel in die Kapselung einbezieht.

Wenn Sie die `snapshot.ini`-Datei nicht anpassen, lädt der Snapshot-Prozess die Datei von einem dieser Speicherorte:

- `Anwendungsdaten\Thinapp\snapshot.ini`

  Dieser Speicherort ist das `AppData`-Verzeichnis der Benutzer.

- `C:\Programme\VMware\VMWare Thinapp\snapshot.ini`

  Dies ist der Speicherort, von dem aus ThinApp das Dienstprogramm `snapshot.exe` ausführt.

Virtuelles Dateisystem ThinApp

D

ThinApp speichert die Unterschiede zwischen Snapshots während des Setup Capture-Prozesses in einem virtuellen Dateisystem und einer virtuellen Registrierung.

Diese Informationen über das virtuelle Dateisystem beinhalten folgende Themen:

- „Virtuelle Dateisystemformate“ auf Seite 111
- „Zusammengeführte und virtuelle Ansichten des Dateisystems“ auf Seite 111
- „Verwenden von Ordnermakros“ auf Seite 112

Virtuelle Dateisystemformate

ThinApp generiert die folgenden virtuellen Dateisystemformate:

- Build

  Durch den Setup Capture-Prozess wird dieses Format aus Dateien direkt vom physischen Dateisystem generiert. ThinApp verwendet Ordnermakros, um Speicherorte für Windows-Shell-Ordner darzustellen.

- Eingebettet

  Die Datei `build.bat` löst einen Build-Prozess aus, der ein schreibgeschütztes Dateisystem in ausführbare Dateien einbettet. Die ausführbaren Dateien stellen Clientcomputern blockbasiertes Streaming zur Verfügung. ThinApp komprimiert das Dateisystem.

- Sandbox

  Bei Ausführung der gekapselten Anwendung wird die Verzeichnisstruktur mit Lese- und Schreibrechten generiert, in der Dateiinformationen enthalten sind, die von der Anwendung modifiziert werden. Zu den Dateimodifikationen, die ThinApp zum Extrahieren eingebetteter virtueller Dateien in die Sandbox veranlassen, gehören folgende Vorgänge:

  - Änderung des Zeitstempels oder der Attribute einer Datei
  - Öffnen einer Datei mit Schreibzugriff
  - Abschneiden einer Datei
  - Umbenennen oder Verschieben einer Datei

Eingebettete und Sandbox-Dateisysteme verwenden Ordnermakros, damit Dateipfade dynamisch zur Laufzeit erweitert werden können.

Zusammengeführte und virtuelle Ansichten des Dateisystems

Isolationsmodi legen fest, ob ThinApp die Anwendung mit einer zusammengeführten Anzeige des virtuellen und physischen Dateisystems oder einer Anzeige der virtuellen Dateien bereitstellt. Informationen über Isolationsmodi erhalten Sie unter „Ändern der Isolationsmodi“ auf Seite 24.

Verwenden von Ordnermakros

ThinApp verwendet Makros, um Dateisystempfade darzustellen, die sich ändern können, wenn virtualisierte Anwendungen auf verschiedenen Windows-Betriebssystemen oder Computern laufen. Durch die Verwendung von Makros wird es möglich, die gemeinsam genutzten Profilinformationen einer Anwendung sofort in verschiedene Betriebssysteme zu migrieren.

Sie können beispielsweise eine Anwendung auf einem System kapseln, das `C:\WINNT` als Windows-Verzeichnis verwendet, und die Anwendung auf einem System mit `C:\Windows` als Windows-Verzeichnis bereitstellen. ThinApp wandelt `C:\WINNT` während der Kapselung dieses Systems in `%SystemRoot%` um und erweitert während der Laufzeit dieses Systems `%SystemRoot%` zu `C:\Windows`.

Registriert eine Anwendung DLLs auf `C:\winnt\system32`, während sie unter Windows 2000 ausgeführt wird, kann der Benutzer die Anwendung schließen und sich auf einem Computer mit dem Betriebssystem Windows XP anmelden. Auf dem Computer mit Windows XP sind die Dateien scheinbar unter `C:\Windows\system32` vorhanden und alle zugehörigen Registrierungsschlüssel verweisen auf `C:\windows\system32`.

Unter Windows Vista verschiebt ThinApp Windows SxS DLLs und Richtlinieninformationen und passt sie an Windows Vista an, anstatt Dateipfadstile von Windows XP zu verwenden. Diese Funktion ermöglicht es den meisten Anwendungen, auf aktualisierte oder ältere Betriebssysteme zu migrieren.

ThinApp bietet SxS-Unterstützung für Anwendungen die unter Windows 2000 ausgeführt werden, auch wenn das Basisbetriebssystem dies nicht tut. Diese Unterstützung ermöglicht den meisten in Windows XP gekapselten Anwendungen die unveränderte Ausführung unter Windows 2000.

Liste der Ordnermakros

ThinApp verwendet die Datei `shfolder.dll`, um den Speicherort von Shell-Ordnern zu ermitteln. Ältere Versionen der Datei `shfolder.dll` bieten keine Unterstützung für einige Makronamen.

Makros, für die `shfolder.dll`-Version 5.0 oder höher erforderlich ist, sind unter anderem `%ProgramFilesDir%`, `%Common AppData%`, `%Local AppData%`, `%My Pictures%` und `%Profile%`.

Makros, für die `shfolder.dll`-Version 6.0 oder höher erforderlich ist, sind unter anderem `%My Videos%`, `%Personal%` und `%Profiles%`.

Tabelle D-1 listet die verfügbaren Ordnermakros auf.

Tabelle D-1. Ordnermakros

Makroname	Typischer Speicherort
%AdminTools%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Startmenü\Programme\Verwaltung
%AppData%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Anwendungsdaten
%CDBurn Area%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Lokale Einstellungen\Anwendungsdaten\Microsoft\CD Burning
%Common AdminTools%	C:\Dokumente und Einstellungen\All Users\Startmenü\Programme\Verwaltung
%Common AppData%	C:\Dokumente und Einstellungen\All Users\Anwendungsdaten
%Common Desktop%	C:\Dokumente und Einstellungen\All Users\Desktop
%Common Documents%	C:\Dokumente und Einstellungen\All Users\Dokumente
%Common Favorites%	C:\Dokumente und Einstellungen\All Users\Favoriten
%Common Programs%	C:\Dokumente und Einstellungen\All Users\Startmenü\Programme
%Common StartMenu%	C:\Dokumente und Einstellungen\All Users\Startmenü
%Common Startup%	C:\Dokumente und Einstellungen\All Users\Startmenü\Programme\Autostart
%Common Templates%	C:\Dokumente und Einstellungen\All Users\Vorlagen
%Cookies%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Cookies

Tabelle D-1. Ordnermakros (Fortsetzung)

Makroname	Typischer Speicherort
%Desktop%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Desktop
%Drive_c%	C:\
%Drive_m%	M:\
%Favorites%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Favoriten
%Fonts%	C:\Windows\Fonts
%History%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Lokale Einstellungen\Verlauf
%Internet Cache%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Lokale Einstellungen\Temporary Internet Files
%Local AppData%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Lokale Einstellungen\Anwendungsdaten
%My Pictures%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Eigene Dateien\Eigene Bilder
%My Videos%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Eigene Dateien\Eigene Videos
%NetHood%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Netzwerkumgebung
%Personal%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Eigene Dateien
%PrintHood%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Druckumgebung
%Profile%	C:\Dokumente und Einstellungen\<Benutzer_Name>
%Profiles%	C:\Dokumente und Einstellungen
%Program Files Common%	C:\Programme\Gemeinsame Dateien
%ProgramFilesDir%	C:\Programme
%Programs%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Startmenü\Programme
%Recent%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Zuletzt verwendete Dokumente
%Resources%	C:\Windows\Resources
%Resources Localized%	C:\Windows\Resources\<Sprache_ID>
%SendTo%	C:\Dokumente und Einstellungen\<Benutzer_Name>\SendTo
%Startup%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Startmenü\Programme\Autostart
%SystemRoot%	C:\Windows
%SystemSystem%	C:\Windows\System32
%TEMP%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Lokale Einstellungen\Temp
%Templates%	C:\Dokumente und Einstellungen\<Benutzer_Name>\Vorlagen

Verarbeitung von %SystemRoot%

Eine Terminaldienste-Umgebung verfügt über ein freigegebenes Windows-Verzeichnis, wie beispielsweise `C:\Windows`, und ein privates Windows-Verzeichnis, wie beispielsweise `C:\Dokumente und Einstellungen\User\Windows`. In dieser Umgebung verwendet ThinApp das benutzerspezifische Verzeichnis für `%SystemRoot%`.

ThinApp-Skripts

E

Skripts ändern das Verhalten von virtuellen Anwendungen dynamisch. Sie können vor dem Starten einer mit ThinApp paketierten Anwendung oder nach dem Beenden der Anwendung einen benutzerdefinierten Code erstellen. Sie können Skripts zum Authentifizieren der Benutzer und zum Laden der Konfigurationsdateien von einer physischen in eine virtuelle Umgebung verwenden.

Rückruffunktionen führen während bestimmter Ereignisse Codes aus. Wenn die Anwendungen untergeordnete Prozesse erstellen, verwenden Sie Rückruffunktionen, um den Code nur im übergeordneten Hauptprozess auszuführen.

API-Funktionen führen ThinApp-Funktionen aus und interagieren mit der ThinApp-Laufzeit. API-Funktionen können Benutzer authentifizieren und den Start von Anwendungen für nicht autorisierte Benutzer verhindern.

Das Hinzufügen von Skripts zu Ihrer Anwendung beinhaltet das Erstellen einer ANSI-Textdatei mit der Dateierweiterung `.vbs` im Projektverzeichnis der Stammanwendung. Das Stammprojektverzeichnis ist das gleiche Verzeichnis, das die Datei `Package.ini` enthält. Während des Build-Prozess fügt ThinApp die Skriptdateien zu der ausführbaren Datei hinzu und führt jede der Skriptdateien während der Laufzeit aus.

ThinApp verwendet VBScript zum Ausführen von Skriptdateien. Informationen über VBScript finden Sie in der Dokumentation für Microsoft VBScript. Sie können VBScript zum Zugriff auf die COM-Steuerelemente, die auf dem Hostsystem oder in dem virtuellen Paket registriert sind, verwenden.

Dieser Abschnitt umfasst die folgenden Themen:

- „Rückruffunktionen“ auf Seite 115
- „Verwenden der Skripts in einer ThinApp-Umgebung“ auf Seite 116
- „API-Funktionen“ auf Seite 119

Rückruffunktionen

Rückruffunktionen mit bestimmten Namen können nur unter bestimmten Bedingungen ausgeführt werden. So können beispielsweise Rückruffunktionen nur dann einen Skriptcode ausführen, wenn eine Anwendung gestartet oder beendet wird.

Namen von Rückruffunktionen:

- `OnFirstSandboxOwner` – Wird nur aufgerufen, wenn eine Anwendung zum ersten Mal die Sandbox sperrt. Dieser Rückruf wird nicht aufgerufen, wenn eine zweite Kopie der gleichen Anwendung die gleiche Sandbox verwendet, während die erste Kopie ausgeführt wird. Wenn die erste Anwendung einen Unterprozess startet und beendet wird, sperrt der zweite Unterprozess die Sandbox und verhindert, dass dieser Rückruf ausgeführt wird, bis alle Unterprozesse beendet wurden und die Anwendung erneut ausgeführt wird.
- `OnFirstParentStart` – Wird aufgerufen, bevor eine ausführbare ThinApp-Datei ausgeführt wird, unabhängig davon, ob die Sandbox gleichzeitig von einer anderen gekapselten ausführbaren Datei verwendet wird.

- `OnFirstParentExit` – Wird aufgerufen, wenn der erste übergeordnete Prozess beendet ist. Wenn ein übergeordneter Prozess einen untergeordneten Prozess ausführt und beendet wird, wird dieser Rückruf auch dann aufgerufen, wenn der untergeordnete Prozess weiterhin ausgeführt wird.
- `OnLastProcessExit` – Wird aufgerufen, wenn der letzte Prozess, der von der Sandbox ausgeführt wird, beendet wird. Falls ein übergeordneter Prozess einen untergeordneten Prozess ausführt und beendet wird, wird dieser Rückruf aufgerufen, wenn der untergeordnete Prozess beendet wird.

Das folgende Beispiel für einen Rückruf zeigt die Funktionen für `OnFirstSandboxOwner` und `OnFirstParentExit`.

```
------------------------example.vbs ---------------------------------
Function OnFirstSandboxOwner
Msgbox "The sandbox owner is: " + GetCurrentProcessName
End Function

Function OnFirstParentExit
msgbox "Quiting application: " + GetCurrentProcessName
End Function

msgbox "This code will execute for all parent and child processes"
---------------------------------------------------------------------
```

Verwenden der Skripts in einer ThinApp-Umgebung

Unter folgenden Umständen kann der Einsatz eines Skripts sinnvoll sein:

- Timeout einer Anwendung an einem bestimmten Datum.
- Ausführen einer `.bat`-Datei von einer Netzwerkfreigabe innerhalb der virtuellen Umgebung.
- Ändern der virtuellen Registrierung.
- Importieren der `.reg`-Datei während der Laufzeit.
- Anhalten eines virtuellen Dienstes, wenn die Hauptanwendung beendet wird.
- Kopieren einer externen Systemkonfigurationsdatei in die virtuelle Umgebung beim Start.

So verwenden Sie ein Skript

1. Speichern Sie den Skriptinhalt in eine Nur-Text-Datei mit der `.vbs`-Erweiterung und legen Sie sie im gleichen Verzeichnis wie Ihre `Package.ini`-Datei ab.

   Sie können einen beliebigen Dateinamen verwenden. ThinApp fügt bei der Build-Zeit des Pakets alle `.vbs`-Dateien zum Paket hinzu.
2. Erstellen Sie die Anwendung erneut.

Beispiel: .bat

Das folgende Skript führt eine externe `.bat`-Datei von einer Netzwerkfreigabe innerhalb der virtuellen Umgebung aus. Die `.bat`-Datei führt Änderungen an der virtuellen Umgebung durch, indem sie Dateien kopiert, Dateien löscht oder mithilfe von `regedit /s regfile.reg` Registrierungsänderungen durchführt. Führen Sie dieses Skript nur für den ersten übergeordneten Prozess aus. Wenn Sie dieses Skript für andere Prozesse ausführen, führt jede Kopie des Dienstprogramms `cmd.exe` das Skript aus und löst eine unendliche Rekursion aus.

```
Function OnFirstParentStart
 Set Shell = CreateObject("Wscript.Shell")
 Shell.Run "\\jcdesk2\test\test.bat"
End Function
```

Beispiel: Timeout

Das folgende Skript verhindert, dass eine Anwendung nach einem bestimmten Datum weiterhin ausgeführt wird. Das VBS-Datum verwendet unabhängig vom Gebietsschema das Format `#mm/dd/yyyy#`.

Diese Überprüfung wird beim Start des übergeordneten sowie beim Start aller untergeordneten Prozesse durchgeführt.

```
if Date >= #03/20/2007# then
 msgbox "This application has expired, please contact Administrator"
 ExitProcess 0
end if
```

Ändern der virtuellen Registrierung

Die folgende Skriptprozedur ändert die virtuelle Registrierung während der Laufzeit, um einen externen ODBC-Treiber von dem gleichen Verzeichnis zu laden, in dem sich die ausführbare Paketdatei befindet.

So ändern Sie die Registrierung

1 Suchen Sie den Pfad zu den ausführbaren Paketdateien.

```
Origin = GetEnvironmentVariable("TS_ORIGIN")
```

2 Suchen Sie den letzten Slash in dem Pfad und kopieren Sie die Zeichen vor dem Slash.

```
LastSlash = InStrRev(Origin, "\")
SourcePath = Left(Origin, LastSlash)
```

3 Bilden Sie einen neuen Pfad zu der `ODBC DLL`-Datei, der sich außerhalb des Pakets befindet.

```
DriverPath=SourcePath + "tsodbc32.dll"
```

4 Ändern Sie die virtuelle Registrierung, so dass sie auf diesen Speicherort zeigt.

```
Set WSHShell = CreateObject("Wscript.Shell")
WSHShell.RegWrite "HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBCINST.INI\Transoft ODBC
Driver\Driver," DriverPath
```

Diese Änderung führt dazu, dass die Anwendung die DLL von einem externen Speicherort aus lädt.

Beispiel: .reg

Mit dem folgenden Skript werden die Registrierungswerte von einer externen `.reg`-Datei während der Laufzeit in die virtuelle Registrierung importiert.

```
Function OnFirstParentStart
ExecuteVirtualProcess "regedit /s c:\tmp\somereg.reg"
End Function
```

Beispiel: Anhalten eines Dienstes

Mit dem folgenden Skript wird ein virtueller oder systemeigener Dienst angehalten, wenn die Hauptanwendung beendet wird.

```
Function OnFirstParentExit
Set WshShell = CreateObject("WScript.Shell")
WshShell.Run "net stop ""iPod Service"""
End Function
```

Beispiel: Kopieren einer Datei

In den folgenden Skriptabschnitten wird gezeigt, wie eine Konfigurationsdatei, die sich im gleichen Verzeichnis befindet wie die gekapselte ausführbare Datei, bei jedem Anwendungsstart in das virtuelle Dateisystem kopiert wird. Dieses Skript ist nützlich für eine externe Konfigurationsdatei, die nach der Bereitstellung einfach zu bearbeiten ist. Da die Kopierfunktion jedes Mal ausgeführt wird, wenn Sie die Anwendung ausführen, werden alle Änderungen an der externen Version in der virtuellen Version reflektiert.

Wenn beispielsweise Ihre gekapselte ausführbare Datei von dem Speicherort `\\server\share\myapp.exe` ausgeführt wird, sucht dieses Skript nach einer Konfigurationsdatei, die sich am Speicherort `\\server\share\config.ini` befindet und kopiert sie in den Speicherort des virtuellen Dateisystems `c:\Programme\Meine Anwendung\config.ini`.

Indem Sie diesen Code in die Funktion `OnFirstParentStart` einfügen, wird sie nur einmal pro Ausführung des Skripts aufgerufen. Ansonsten wird sie für jeden untergeordneten Prozess ausgeführt.

```
Function OnFirstParentStart
```

ThinApp richtet `TS_ORIGIN` ein, um den vollständigen Pfad zu einem gekapselten ausführbaren Dateipaket anzugeben. Eine virtuelle Anwendung richtet die Variable `TS_ORIGIN` auf dem physischen Pfad des primären Datencontainers ein. Wenn Sie eine virtuelle Anwendung haben, die aus `main.exe` und `shortcut.exe` besteht, befinden sich beide Dateien in `C:\VirtApp`. Wenn Sie die Datei `main.exe` ausführen, wird `TS_ORIGIN var` auf `C:\VirtApp\main.exe` eingerichtet. Wenn Sie die Datei `shortcut.exe` ausführen, wird die Umgebungsvariable `TS_ORIGIN` auf `C:\VirtApp\main.exe` eingerichtet. Die Umgebungsvariable wird immer auf den primären Datencontainer eingerichtet, auch wenn Sie eine Verknüpfung erstellen. Wenn Sie VBScripts ausführen, die in dem Paket enthalten sind, ist die Variable bereits eingerichtet und für die Skripts verfügbar.

```
Origin = GetEnvironmentVariable("TS_ORIGIN")
```

Sie können den Dateinamen von `TS_ORIGIN` trennen, indem Sie nach dem letzten Backslash suchen und alle Zeichen nach diesem Backslash entfernen.

```
LastSlash = InStrRev(Origin, "\")
SourcePath = Left(Origin, LastSlash)
```

Die Quelldatei, die in die virtuelle Umgebung zu kopieren ist, ist der Pfad für das Paket plus `config.ini`.

```
SourceFile = SourcePath + "Config.ini"
```

Der zu kopierende Speicherort kann für unterschiedliche Computer unterschiedlich sein, wenn das Verzeichnis `Programme` einem anderem Speicherort als `c:\` zugeordnet wurde. Mit dem folgenden Aufruf kann ThinApp ein Makro ausführen, um den korrekten Speicherort für den lokalen Computer aufzufinden.

```
DestFile = ExpandPath("%ProgramFilesDir%\MyApplication\Config.ini")
```

Verwenden Sie den Dateiparameter `systemObject`, um zu überprüfen, ob die Quelldatei vorhanden ist.

```
Set objFSO = CreateObject("Scripting.filesystemObject")
If objFSO.FileExists(SourceFile) Then
```

Wenn die Quelldatei vorhanden ist, kopieren Sie sie in das virtuelle Dateisystem. Das virtuelle Verzeichnis `%ProgramFilesDir%\MyApplication` befindet sich im Paket.

```
objFSO.CopyFile SourceFile, DestFile, TRUE
End if
End Function
```

Hinzufügen eines Wertes zur Systemregistrierung

Mit dieser Skriptprozedur wird ein Wert zu der physischen Systemregistrierung hinzugefügt.

So fügen Sie einen Wert zur Systemregistrierung hinzu

1 Erstellen Sie eine `.reg`-Datei und führen Sie den Befehl `regedit /s` als externen Prozess aus, der auf die Systemregistrierung anstelle der virtuellen Registrierung zugreift.

```
Function OnFirstParentStart
```

2 Erstellen Sie die `.reg`-Datei an einem Speicherort, bei dem der Parameter `IsolationMode` auf `Merged (Zusammengeführt)` eingerichtet ist, damit die virtuelle Umgebung mit diesem Skript auf sie zugreifen kann und die physische Umgebung mit dem Befehl `regedit /s`.

```
RegFileName = ExpandPath("%Personal%\thin.reg")
Set fso = CreateObject("Scripting.filesystemObject")
Set RegFile = fso.CreateTextFile(RegFileName, true)
```

Das Verzeichnis %Personal% ist ein Verzeichnis, das standardmäßig den Isolationsmodus „Merged (Zusammengeführt)“ besitzt.

3 Erstellen Sie die .reg-Datei.

```
RegFile.WriteLine("Windows Registry Editor Version 5.00")
RegFile.WriteBlankLines(1)
RegFile.WriteLine("[HKEY_CURRENT_USER\Software\Thinapp\demo]") RegFile.WriteLine(chr(34) and
"InventoryName" and chr(34) and "=" and chr(34) and GetBuildOption("InventoryName") and
chr(34))
RegFile.Close
```

4 Geben Sie die Informationen in die Systemregistrierung ein.

```
RegEditPid = ExecuteExternalProcess("regedit /s " and chr(34) and RegFileName and chr(34))
WaitForProcess RegEditPid, 0
```

Warten Sie, bis der Prozess beendet ist.

5 Bereinigen Sie die Umgebung.

```
fso.DeleteFile(RegFileName)
End Function
```

API-Funktionen

Sie können API-Funktionen verwenden, mit denen ThinApp zum Fertigstellen von Vorgängen wie dem Laden von DLLs wie virtuellen DLLs, zum Konvertieren von Pfaden vom Makroformat zum Systemformat und zum Ausführen von Befehlen innerhalb der virtuellen Umgebung angewiesen wird.

AddForcedVirtualLoadPath

Mit der Funktion AddForcedVirtualLoadPath(Path) wird ThinApp angewiesen, alle DLLs von dem festgelegten Pfad als virtuelle DLLs zu laden, auch wenn sie sich nicht im Paket befinden.

Verwenden Sie diese Funktion, wenn die Anwendung externe DLLs laden muss, die von DLLs innerhalb des Pakets abhängen.

Parameter

Path

[in] Der Dateiname oder Pfad für DLLs, die als virtuelle DLLs geladen werden sollen.

Beispiele

Sie können alle DLLs als virtuelle DLLs laden, die sich im gleichen Verzeichnis wie die ausführbare Datei befinden.

```
Origin = GetEnvironmentVariable("TS_ORIGIN")
```

TS_ORIGIN ist der Pfad, von dem aus die ausführbare Datei ausgeführt wird.

Sie können den Dateinamen von TS_ORIGIN löschen, indem Sie nach dem letzten Backslash suchen und alle Zeichen nach diesem Backslash entfernen.

```
LastSlash = InStrRev(Origin, "\")
SourcePath = Left(Origin, LastSlash)
```

Sie können ThinApp anweisen, alle DLLs im gleichen oder einem niedrigeren Verzeichnis zu laden, in dem sich die ausführbare Quelldatei befindet.

```
AddForcedVirtualLoadPath(SourcePath)
```

Dieser Prozess ermöglicht Ihnen, zusätzliche Dateien in der SourcePath-Struktur abzulegen, damit sie Importvorgänge für virtuelle DLLs auflösen.

ExitProcess

Die Funktion `ExitProcessExitCode` beendet den aktuellen Prozess und setzt den spezifizierten Fehlercode.

Parameter

`ExitCode`

[in] Der zu setzende Fehlercode. Diese Informationen könnten für einen übergeordneten Prozess zur Verfügung stehen. Ein Wert 0 gibt an, dass kein Fehler vorliegt.

Beispiele

Sie können den Prozess beenden und den Erfolg anzeigen.

```
ExitProcess 0
```

Wenn der Prozess beendet ist, empfängt das Skriptsystem den Rückruf für die Funktion `OnLastProcessExist`. Alle geladenen DLLs führen einen Code für das Beenden aus, um die Umgebung zu bereinigen.

ExpandPath

Die Funktion `ExpandPath(InputPath)` konvertiert einen Pfad vom Makroformat zum Systemformat.

Parameter

`InputPath`

[in] Ein Pfad im Makroformat.

Ausgabe

Der erweiterte Makropfad im Systemformat.

Beispiele

Path = `ExpandPath("%ProgramFilesDir%\Myapp.exe")`

Path = `c:\Programme\myapp.exe`

Alle Makropfade müssen die Zeichen % und # durch #25 und #23 ersetzen.

Path = `ExpandPath("%ProgramFilesDir%\FilenameWithPercent#25.exe")`

Dies wird erweitert auf `C:\Programme\FileNameWithPercent%.exe`.

ExecuteExternalProcess

Die Funktion `ExecuteExternalProcess(CommandLine)` führt einen Befehl außerhalb der virtuellen Umgebung aus. Sie können diese Funktion verwenden, um Änderungen am physischen System vorzunehmen.

Parameter

`CommandLine`

[in] Repräsentation der Anwendung und der Befehlszeilenparameter zur Ausführung außerhalb der virtuellen Umgebung.

Ausgabe

Ganzzahlige Prozess-ID. Sie können die Prozess-ID mit der Funktion `WaitForProcess` verwenden. Siehe „WaitForProcess“ auf Seite 125.

Beispiele

```
ExecuteExternalProcess("cmd.exe /c copy c:\systemfile.txt c:\newsystemfile.txt")
```

Sie können einen Befehl ausführen, der Anführungszeichen in der Befehlszeile erfordert.

```
ExecuteExternalProcess("regsvr32 /s " and chr(34) and "c:\Programme\my.ocx" and chr(34))
```

ExecuteVirtualProcess

Die Funktion `ExecuteVirtualProcess(CommandLine)` führt einen Befehl innerhalb der virtuellen Umgebung aus. Sie können diese Funktion verwenden, um Änderungen an der virtuellen Umgebung vorzunehmen.

Parameter

`CommandLine`

[in] Repräsentation der Anwendung und der Befehlszeilenparameter zur Ausführung außerhalb der virtuellen Umgebung.

Ausgabe

Ganzzahlige Prozess-ID. Sie können die Prozess-ID mit der Funktion `WaitForProcess` verwenden. Siehe „WaitForProcess“ auf Seite 125.

Beispiele

```
ExecuteVirtualProcess("cmd.exe /c copy c:\systemfile.txt c:\virtualfile.txt")
```

Sie können einen Befehl ausführen, der Anführungszeichen in der Befehlszeile erfordert.

```
ExecuteVirtualProcess("regsvr32 /s " and chr(34) and "c:\Programme\my.ocx" and chr(34))
```

GetBuildOption

Die Funktion `GetBuildOption(OptionName)` gibt den Wert einer Einstellung aus, die im Abschnitt `[BuildOptions]` der Datei `Package.ini`, die für gekapselte Anwendungen verwendet wird, festgelegt wird.

Parameter

`OptionName`

[in] Name der Einstellung.

Ausgabe

Diese Funktion gibt einen Zeichenfolgenwert aus. Wenn der angeforderte Optionsname nicht existiert, gibt die Funktion einen leeren Zeichenfolgenwert aus ("").

Beispiele

```
Package.ini enthält:
[BuildOptions]
CapturedUsingVersion=4.0.1-2866
```

In einer VBS-Datei wird die folgende Zeile angezeigt:

```
Value = GetBuildOption("CapturedUsingVersion")
```

GetFileVersionValue

Die Funktion `GetFileVersionValue(Filename, Value)` gibt den Wert für die Versionsinformationen von Dateien wie bestimmten DLL, OCX oder ausführbaren Dateien aus. Sie können diese Funktion dazu verwenden, die interne Versionsnummer einer DLL zu bestimmen oder DLL-Informationen über den Copyright-Inhaber oder einen Produktnamen abzurufen.

Parameter

`Filename`

[in] Der Name des Dateinamens, dessen Versionsinformationen abgerufen werden.

Value

[in] Der Name des Wertes, der von dem Versionsinformationsabschnitt der bestimmten Datei abgerufen werden soll.

Sie können von den meisten DLLs die folgenden Werte abrufen:

- Comments
- InternalName
- ProductName
- CompanyName
- LegalCopyright
- ProductVersion
- FileDescription
- LegalTrademarks
- PrivateBuild
- FileVersion
- OriginalFilename
- SpecialBuild

Ausgabe

Diese Funktion gibt einen Zeichenfolgenwert aus. Wenn der angeforderte Dateiname nicht vorhanden ist oder die Funktion den bestimmten Wert nicht in der Datei finden kann, gibt die Funktion eine leere Zeichenfolge aus ("").

Beispiele

```
FileVersion = GetFileVersionValue("c:\windows\system32\kernel32.dll," "FileVersion")

if FileVersion = "1.0.0.0" then
                MsgBox "This is Version 1.0!"

End if
```

GetCommandLine

Mit der Funktion `GetCommandLine` wird auf die Befehlszeilenparameter zugegriffen, die an das laufende Programm übergeben werden.

Ausgabe

Diese Funktion gibt eine Zeichenfolge aus, die die Befehlszeilenargumente darstellt, die an das derzeit laufende Programm übergeben werden, einschließlich der ursprünglichen ausführbaren Datei.

Beispiele

```
MsgBox "The command line for this EXE was " + GetCommandLine
```

GetCurrentProcessName

Mit der Funktion `GetCurrentProcessName` wird auf den vollständigen virtuellen Pfadnamen des derzeitigen Prozesses zugegriffen.

Ausgabe

Diese Funktion gibt eine Zeichenfolge aus, die den vollständigen ausführbaren Pfadnamen innerhalb der virtuellen Umgebung darstellt. Unter den meisten Umständen lautet dieser Pfad `c:\Programme\...`, auch wenn die Paketquelle von einer Netzwerkfreigabe ausgeführt wird.

Beispiele

```
MsgBox "Running EXE path is " + GetCurrentProcessName
```

GetOSVersion

Die Funktion `GetOSVersion()` gibt Informationen über die aktuelle Windows-Version aus.

Parameter

Diese Funktion hat keine Parameter.

Ausgabe

Diese Funktion gibt eine Zeichenfolge in folgendem Format aus: `MAJOR.MINOR.BUILD_NUMBER.PLATFORM_ID OS_STRING`.

`MAJOR` kann einen der folgenden Werte annehmen:

Windows Vista	6
Windows Server 2008	6
Windows Server 2003	5
Windows XP	5
Windows 2000	5
Windows NT 4.0	4

`MINOR` kann einen der folgenden Werte annehmen:

Windows Vista	0
Windows Server 2008	0
Windows Server 2003	2
Windows XP	1
Windows 2000	0
Windows NT 4.0	0
Windows NT 3.51	51

`BUILD_NUMBER` ist die Versionsnummer des Betriebssystems.

PLATFORM_ID weist einen der folgenden Werte zu:

- `Value = 1` für Windows Me, Windows 98 oder Windows 95 (Windows 95-basiertes Betriebssystem)
- `Value = 2` für Windows Server 2003, Windows XP, Windows 2000 oder Windows NT. (Windows NT-basiertes Betriebssystem)

OS_STRING stellt die Informationen über das Betriebssystem dar, wie beispielsweise Service Pack 2.

Beispiele

```
if GetOSVersion() = "5.1.0.2 Service Pack 2"
               then MsgBox "You are running on Windows XP Service Pack 2!"
endif
```

GetEnvironmentVariable

Die Funktion GetEnvironmentVariable(Name) gibt die Umgebungsvariable aus, die der Variablen Name zugewiesen wurde.

Parameter

Name

[in] Der Name der Umgebungsvariable, für die der Wert abgerufen wird.

Ausgabe

Diese Funktion gibt den Zeichenfolgenwert aus, der der Umgebungsvariable Name zugewiesen wurde.

Beispiele

```
MsgBbox "The package source EXE is " + GetEnvironmentVariable("TS_ORIGIN")
```

RemoveSandboxOnExit

Mit der Funktion RemoveSandboxOnExit(YesNo) kann durch eine Ja/Nein-Auswahl bestimmt werden, ob die Sandbox gelöscht wird, wenn der letzte untergeordnete Prozess beendet wird.

Wenn Sie den Parameter RemoveSandboxOnExit in der Datei Package.ini auf 1 setzen, lautet das Standard-Bereinigungsverhalten für das Paket Yes (Ja). Sie können das Bereinigungsverhalten auf No (Nein) einrichten, indem Sie RemoveSandboxOnExit mit dem Wert 0 aufrufen. Wenn Sie den Eintrag RemoveSandboxOnExit=1 in der Datei Package.ini nicht ändern, lautet das Standard-Bereinigungsverhalten für das Paket No (Nein). Sie können das Bereinigungsverhalten zu Yes (Ja) ändern, indem Sie RemoveSandboxOnExit mit dem Wert 1 aufrufen.

Parameter

Yes No

[in] Wollen Sie nach dem Ende des letzten Prozesses eine Bereinigung durchführen? 1=Yes (Ja), 0=No (Nein)

Beispiele

Das folgende Beispiel aktiviert die Bereinigung.

```
RemoveSandboxOnExit 1
```

Das folgende Beispiel deaktiviert die Bereinigung.

```
RemoveSandboxOnExit 0
```

SetEnvironmentVariable

Die Funktion SetEnvironmentVariable(Name, Value) richtet den Wert einer Umgebungsvariable ein.

Parameter

Name

[in] Der Name der Umgebungsvariable zum Speichern des Wertes.

Value

[in] Der zu speichernde Wert.

Beispiele

```
SetEnvironmentVariable "PATH", "C:\Windows\system32"
```

SetfileSystemIsolation

Die Funktion `Setfile systemIsolation(Directory, IsolationMode)` richtet den Isolationsmodus eines Verzeichnisses ein.

Parameter

`Directory`

[in] Vollständiger Pfad des Verzeichnisses, dessen Isolationsmodus eingerichtet werden soll.

`IsolationMode`

[in] Einzurichtender Isolationsmodus.

1 = WriteCopy
2 = Merged
3 = Full

Beispiele

Sie können den Isolationsmodus „Merged (Zusammengeführt)“ für das temporäre Verzeichnis einrichten.

```
Setfile systemIsolation GetEnvironmentVariable("TEMP"), 2
```

SetRegistryIsolation

Die Funktion `SetRegistryIsolation(RegistryKey, IsolationMode)` richtet den Isolationsmodus eines Registrierungsschlüssels ein.

Parameter

`RegistryKey`

[in] Der Registrierungsschlüssel, auf den der Isolationsmodus eingerichtet werden soll. Beginnen Sie mit `HKLM` für `HKEY_LOCAL_MACHINE`, `HKCU` für `HKEY_CURRENT_USER` und `HKCR` für `HKEY_CLASSES_ROOT`.

`IsolationMode`

[in] Einzurichtender Isolationsmodus.

1 = WriteCopy
2 = Merged
3 = Full

Beispiele

Sie können den Isolationsmodus „Full (Voll)“ für `HKEY_CURRENT_USER\Software\Thinapp\Test` einrichten.

```
SetRegistryIsolation "HKCU\Software\Thinapp\Test," 3
```

WaitForProcess

Die Funktion `WaitForProcess(ProcessID, TimeOutInMilliSeconds)` wartet, bis die Prozess-ID die Ausführung beendet hat.

Parameter

`ProcessID`

[in] Die zu beendende Prozess-ID. Die Prozess-ID kommt entweder von `ExecuteExternalProcess` oder von `ExecuteVirtualProcess`.

`TimeOutInMilliSeconds`

[in] Die maximale Zeitspanne, die auf das Beenden des Prozesses gewartet wird, bis der Vorgang fortgesetzt wird. Ein Wert 0 legt INFINITE (unendlich) fest.

Ausgabe

Diese Funktion gibt eine Ganzzahl aus.

0 = Timeout fails (Timeout fehlgeschlagen)
1 = Process exits (Prozess ist beendet)
2 = Process does not exist or security is denied (Prozess ist nicht vorhanden oder Sicherheit ist verweigert)

Beispiele

```
id = ExecuteExternalProcess("cmd.exe")
WaitForProcess(id, 0)
```

Glossar

Application Link
Ein Dienstprogramm, das voneinander abhängige Anwendungen während der Laufzeit mit einer Basisanwendung verknüpft und diese gemeinsam startet, sobald die Basisanwendung gestartet wird. Sie können das Dienstprogramm verwenden, um Komponentenpakete einzeln bereitzustellen und zu aktualisieren, anstatt alle Komponenten im selben Paket zu kapseln.

Application Sync
Ein Dienstprogramm zur Aktualisierung einer Anwendung, das neue verpackte Versionen auf einem Server oder in einem freigegebenem Netzwerk erkennt. Update-Einstellungen, wie die Überprüfung eines Update-Servers in bestimmten Zeitintervallen, können konfiguriert werden. ThinApp erkennt die neueste ausführbare Datei der Anwendung und lädt die Unterschiede herunter.

attributes.ini
Diese Datei wendet Konfigurationseinstellungen auf der Verzeichnisebene des Pakets statt auf der Ebene des gesamten Pakets an. Die `##Attributes.ini`-Einstellungen überschreiben alle `Package.ini`-Einstellungen.

Bestandsname
Ein Name, den ThinApp zur internen Nachverfolgung der Anwendung verwendet. Der Bestandsname legt den standardmäßigen Projektverzeichnisnamen fest und wird im Dialogfeld **Software** von Windows angezeigt.

Build
So konvertieren Sie ein ThinApp-Projekt in ein Paket. Ein Paket kann mithilfe des Setup Capture-Assistenten oder mit dem Dienstprogramm `build.bat` erstellt werden.

Einstiegspunkt
Eine ausführbare Datei zum Starten der gekapselten Anwendung. Eine Anwendung kann über mehrere Einstiegspunkte verfügen. Zum Beispiel: Die Dateien `Firefox.exe` und `cmd.exe` können als Einstiegspunkte für eine Mozilla Firefox-Anwendung dienen. Die primäre Datencontainerdatei muss als einer der Einstiegspunkte vorhanden sein.

Isolationsmodus
Eine Paketeinstellung, die den Lese- und Schreibzugriff auf die physische Umgebung festlegt. ThinApp verfügt über die Isolationsmodi: WriteCopy, Merged (Zusammengeführt) und Full (Voll).

kapseln
So paketieren Sie eine Anwendung in eine virtuelle Umgebung und legen Sie die anfänglichen Anwendungsparameter fest. ThinApp stellt den Setup Capture-Assistenten oder das Dienstprogramm `snapshot.exe` zur Erstellung eines portablen Anwendungspakets bereit, das unabhängig von dem Betriebssystem ist, auf dem es ausgeführt wird.

logging.dll
Ein Dienstprogramm, das `.trace`-Dateien generiert.

MSI
Ein Windows Installer-Container, der für Anwendungs-Bereitstellungstools nützlich ist. Sie können die gekapselte Anwendung als MSI-Datei anstatt einer ausführbaren Datei verteilen.

Nachüberprüfung
Zum Erstellen einer Abbildung oder eines Snapshots einer Maschine nach der Installation der Anwendung, die Sie kapseln möchten. Der Kapselungsvorgang speichert die Unterschiede zwischen den Vor- und Nachüberprüfungsabbildungen in einem virtuellen Dateisystem und einer virtuellen Registrierung. *Siehe auch* sbmerge.exe, Snapshot.

nativ
Bezieht sich eher auf die physische Umgebung als auf die virtuelle Umgebung. *Siehe auch* physisch.

Netzwerk-Streaming
Bei diesem Vorgang wird ein Paket von einem Zentralserver aus ausgeführt. ThinApp lädt nach Bedarf Blöcke der Anwendung herunter, um eine schnelle Verarbeitung und Anzeige zu gewährleisten.

neu aufgesetzte Maschine
Der Computer oder die virtuelle Maschine, nur mit dem elementaren Windows-Betriebssystem installiert, auf dem die Anwendung erfasst wird. Die Betriebssystemversion von Windows muss die früheste Version sein, mit der eine Anwendung ausgeführt werden soll.

package.ini
Die Datei, die die Konfigurationseinstellungen auf das Paket anwendet und im gekapselten Anwendungsordner gespeichert ist. Der Setup Capture-Assistent richtet die ursprünglichen Konfigurationseinstellungen ein.

Paket
Die virtuellen Anwendungsdateien, die vom ThinApp-Erstellungsvorgang generiert werden. Das Paket enthält die primäre Datencontainerdatei und Einstiegspunkt-Dateien zum Zugriff auf die Anwendung.

physisch
Bezieht sich auf den Computerarbeitsspeicher und das Dateisystem, in dem alle Standardprozesse von Windows ausgeführt werden. Abhängig von den Einstellungen des Isolationsmodus in ThinApp, können Vorgänge in der virtuellen Umgebung auf die physische Umgebung zugreifen. *Siehe auch* nativ, virtuell.

Primärer Datencontainer
Die Hauptdatei zum Zugriff auf die virtuelle Anwendung. Die Datei ist eine `.exe`- oder eine `.dat`-Datei, in der die ThinApp-Laufzeit sowie das schreibgeschützte virtuelle Dateisystem und die virtuelle Registrierung enthalten sind. Der primäre Datencontainer muss im selben `/bin`-Verzeichnis zusammen mit allen untergeordneten ausführbaren Dateien der Anwendung gespeichert werden.

Projekt
Die Daten, die beim Kapselungsvorgang erzeugt werden, bevor ein Paket erstellt wird. Der Kapselungsprozess verwendet den Bestandsnamen als Standardprojektverzeichnisnamen. Sie können die Parameter in den Projektdateien anpassen, bevor Sie ein Anwendungspaket erzeugen. Eine gekapselte Anwendung kann erst bereitgestellt werden, nachdem ein Paket aus dem Projekt erstellt wurde.

Protokoll-Monitor
Ein Dienstprogramm, das die Aktivitäten ausführbarer Dateien, die von der gekapselten Anwendung gestartet werden, chronologisch erfasst. Die Datei `log_monitor.exe` ist ausschließlich mit gekapselten Anwendungen, die dieselbe Version von ThinApp verwenden, kompatibel.

Sandbox
Der physische Systemordner, der während der Laufzeit an der virtuellen Anwendung vorgenommene Benutzeränderungen speichert. Sobald die Anwendung gestartet wird, integriert ThinApp die Änderungen aus der Sandbox. Wenn Sie die Sandbox löschen, führt ThinApp die Anwendung zurück in den gekapselten Zustand. Der Standardspeicherort der Sandbox ist `%APPDATA%\Thinstall\<Benutzer_Name>`.

sbmerge.exe
Ein Dienstprogramm, das inkrementelle Updates der Anwendungen erstellt, beispielsweise die Integrierung eines Plug-Ins oder einer Änderung auf einer Browser-Startseite. Das Dienstprogramm `sbmerge.exe` führt die in der Sandbox aufgezeichneten Laufzeitänderungen zurück in ein ThinApp-Projekt.

Snapshot
Eine Statuserfassung des Dateisystems und der Registrierung von Windows während der Kapselung der Anwendung. Der Setup Capture-Vorgang verwendet das Dienstprogramm `snapshot.exe`, um mit einem Snapshot den Systemzustand vor und nach der Installation der Anwendung zu erfassen und speichert die Unterschiede in einem virtuellen Dateisystem und einer virtuellen Registrierung. *Siehe auch* Nachüberprüfung, sbmerge.exe.

snapshot.exe
Ein Dienstprogramm, das Snapshots des Dateisystems und der Registrierung eines Computers erstellt und die Vor- und Nachüberprüfungsvorgänge während des Kapselungsvorgangs vereinfacht. Nur fortgeschrittene Benutzer, die die ThinApp-Funktionalität in andere Plattformen integrieren, werden dieses Dienstprogramm möglicherweise direkt verwenden. *Siehe auch* Nachüberprüfung, sbmerge.exe, Snapshot.

snapshot.ini
Eine Konfigurationsdatei, die die Verzeichnisse und Unterschlüssel festlegt, die bei der Kapselung einer Anwendung von einem ThinApp-Projekt ausgeschlossen werden sollen. Diese Datei kann für Anwendungen angepasst werden.

template.msi
Eine Vorlage für MSI-Dateien, die Sie anpassen können, um die Bereitstellungsverfahren und -normen des Unternehmens einzuhalten. Zum Beispiel können Registrierungseinstellungen hinzufügt werden, die ThinApp als Bestandteil der Installation zu einem Clientcomputer hinzufügen soll.

thinreg.exe
Ein Dienstprogramm, das Dateitypzuordnungen erstellt, das **Start**-Menü und Verknüpfungen einrichtet und das Öffnen von Dateien vereinfacht. Sie müssen das Dienstprogramm `thinreg.exe` ausführen, um ausführbare Dateien zu registrieren. MSI-Dateien automatisieren den `thinreg.exe`-Registrierungsprozess.

tlink.exe
Ein Dienstprogramm, dass während des Kapselungsvorgangs Schlüsselmodule verknüpft.

vftool.exe
Ein Dienstprogramm, das das virtuelle Dateisystem während des Kapselungsvorgangs kompiliert.

virtuell
Bezieht sich auf die logische Datei und den Arbeitsspeicher, in dem eine gekapselte Anwendung ausgeführt wird. Prozesse in einer physischen Umgebung können nicht auf die virtuelle Umgebung zugreifen. *Siehe auch* physisch.

virtuelle Anwendung
Eine Anwendung, die gekapselt wird, damit sie portabel und unabhängig vom Betriebssystem ist, auf dem sie ausgeführt wird.

virtuelle Registrierung
Die Registrierung, wie sie von der gekapselten Anwendung gesehen wird.

virtuelles Dateisystem
Das Dateisystem, wie es von der gekapselten Anwendung gesehen wird.

Vorüberprüfung
Zum Erstellen einer Baselineabbildung oder eines Snapshots einer Maschine vor der Installation der Anwendung, die Sie kapseln möchten. Der Kapselungsvorgang speichert die Unterschiede zwischen den Vor- und Nachüberprüfungsabbildungen in einem virtuellen Dateisystem und einer virtuellen Registrierung. *Siehe auch* Nachüberprüfung, Snapshot.

vregtool.exe
Ein Dienstprogramm, das die virtuelle Registrierung während des Kapselungsvorgangs kompiliert.

Index

Symbole

##Attributes.ini
- Ändern der Isolationsmodi **25**
- bearbeiten **24**
- Vergleichen mit Package.ini **24**, **66**

A

Active Directory
- Package.ini-Parameter verwenden **34**
- Zugriff auf Anwendungen steuern **33**
- Zugriff für Gruppen autorisieren **18**

Aktualisieren von Anwendungen, Methoden und Überlegungen **41**–**52**

Anwendungen
- aktualisieren **41**
- kapseln **15**
- Streaminganforderungen und -empfehlungen **36**
- Überlegungen zur Sandbox in Bezug auf Upgrade-Vorgänge **52**
- Unterschied zwischen Application Sync und Application Link **41**
- von ThinApp nicht unterstützt **12**
- Zugriff steuern für Active Directory-Gruppen **33**

API-Parameter
- AddForcedVirtualLoadPath **119**
- ExecuteExternalProcess **120**
- ExecuteVirtualProcess **121**
- ExitProcess **120**
- ExpandPath **120**
- GetBuildOption **121**
- GetCommandLine **122**
- GetCurrentProcessName **123**
- GetEnvironmentVariable **124**
- GetFileVersionValue **121**
- GetOSVersion **123**
- RemoveSandboxOnExit **124**
- SetEnvironmentVariable **124**
- SetfileSystemIsolation **125**
- SetRegistryIsolation **125**
- WaitForProcess **125**

Application Link
- Ansicht von **44**
- Auswirkung auf Isolationsmodi **46**
- Datei- und Registrierungskollisionen **47**
- definieren **41**, **43**
- Definition für den Zugriff mit dem PermittedGroups-Parameter **46**
- Einrichten der verschachtelten Links **45**
- erforderliche Links **90**
- optionale Links **91**
- Parameter **89**
- Pfadnamenformate **90**
- Speichern mehrerer Versionen von verknüpften Anwendungen **47**
- Verknüpfen von Paketen mit Basisanwendungen unter Verwendung von Application Sync **48**
- Workflow-Beispiel **44**

Application Sync
- Aktualisieren der thinreg.exe-Registrierungen **43**
- Aktualisieren von Basisanwendungen mit verknüpften Paketen **48**
- Auswirkung auf ausführbare Dateien mit Einstiegspunkt **43**
- Auswirkung auf thinreg.exe **28**
- Bearbeiten von Parametern **42**
- Beibehalten des primären Datencontainernamens **43**
- definieren **41**
- Erzwingen der Updates mit appsync.exe-Befehlen **48**
- Kollidieren mit automatischen Update-Funktionen **41**
- Korrigieren der Updates **43**
- Parameter **92**

Ausschneide- und Einfügevorgänge, ThinApp-Beschränkungen **37**

B

bereitstellen
- Anwendungen auf Netzwerkfreigabe **28**
- Anwendungen mit Bereitstellungstools **27**
- ausführbare Dateien **28**
- MSI-Dateien **27**

Bereitstellungstools, unter Verwendung von MSI-Dateien **27**

Bestandsname, Zweck **18**

Betriebssysteme
- Unterstützung für **11**
- Verwenden der frühesten Version für die ThinApp-Installation **13**

C

cmd.exe, definieren **17**

Computer

Definition von sauberem System **12**
virtuelle Maschine als sauberes System verwenden **13**

D

Datencontainer, *Siehe* primärer Datencontainer
DCOM-Dienste, Zugriff auf gekapselte Anwendungen **12**
DLLs
Aufzeichnung durch Protokoll-Monitor **54**
in den Arbeitsspeicher laden **58**

E

Einstiegspunkte
Aktualisieren mit Application Sync **43**
definieren **17**
zur Problembehandlung **17**

G

Gerätetreiber, nicht kompatibel mit ThinApp **12**
globale Hook-DLLs, beschränkte Funktion mit ThinApp **12**

I

iexplore.exe, definieren **17**
Installation von ThinApp **13**
Isolationsmodi
ändern **24**
Auswirkung auf virtuelles Dateisystem **111**
Beispielkonfiguration **39**
Full (Voll) **24**
Merged (Zusammengeführt) **21**
Verwenden von Application Link **46**
WriteCopy **21**
Isolationsmodus „Merged“ **21**
Isolationsmodus „WriteCopy“ **21**

K

Kapselung von Anwendungen
Bewertung von Anwendungsabhängigkeiten **15**
Empfehlungen vor der **16**
mit dem Dienstprogramm snapshot.exe **107**
mit dem Setup Capture-Assistenten **16**–**23**
Phasen der **15**
Komprimierung
ausführbarer Dateien **22**
für Trace-Dateien **55**

M

Microsoft Vista, Bereitstellung von MSI-Dateien **33**
MSI-Dateien
Automatisieren des Dienstprogramms thinreg.exe **22**
Bereitstellung auf Microsoft Vista **33**
Datenbank aufbauen **31**
generieren **22**
Installationsverzeichnis überschreiben **33**
Package.ini ändern **32**
Parameter **94**
Parameter anpassen **31**

N

Netzwerk, Pakete per Stream übertragen **34**

P

Package.ini
AccessDeniedMsg **71**
Active Directory-Parameter **34**
AddPageExecutePermission **71**
AllowExternalProcessModifications **79**
AllowUnsuppportedExternalChildProcesses **79**
Ändern der Isolationsmodi **25**
AnsiCodePage **85**
AppSyncClearSandboxOnUpdate **92**
AppSyncExpireMessage **92**
AppSyncExpirePeriod **93**
AppSyncUpdateFrequency **93**
AppSyncUpdateMessage **93**
AppSyncURL **93**
AppSyncWarningFrequency **94**
AppSyncWarningMessage **94**
AppSyncWarningPeriod **94**
AutoShutdownServices **80**
AutoStartServices **80**
Bearbeiten von Application Sync-Parametern **42**
Beschreibung allgemeiner Parameter **23**
BlockSize **81**
CachePath **77**
CapturedUsingVersion **84**
ChildProcessEnvironmentDefault **80**
ChildProcessEnvironmentExceptions **81**
CommandLine **86**
CompressionType **82**
DirectoryIsolationMode **68**
Disabled **86**
DisableTracing **84**
ExcludePattern **70**
ExternalCOMObjects **73**
ExternalDLLs **74**
FileTypes **69**
Icon **82**
InventoryName **98**
IsolatedMemoryObjects **74**
IsolatedSynchronizationObjects **75**
Konfigurieren der Application Link-Parameter **89**
Konfigurieren der Build-Parameter **70**
Konfigurieren der Laufzeitparameter **66**
Konfigurieren der Sicherheitsparameter **71**

Konfigurieren der Speicherparameter **77**
Konfigurieren von Application Sync-Parametern **92**
Konfigurieren von Datei- und Protokollzuordnungsparametern **69**
Konfigurieren von einzelnen Anwendungsparametern **86**
Konfigurieren von Größenparametern **81**
Konfigurieren von Isolationsparametern **68**
Konfigurieren von lokalen Parametern **85**
Konfigurieren von MSI-Parametern **94**
Konfigurieren von Objekt- und DLL-Parametern **73**
Konfigurieren von Protokollierungsparametern **83**
Konfigurieren von Prozess- und Dienstparametern **79**
Konfigurieren von Sandbox-Parametern **98**
Konfigurieren von Symbolparametern **82**
Konfigurieren von Versionsparametern **84**
LocaleIdentifier **85**
LocaleName **86**
LogPath **84**
MSIArpProductIcon **94**
MSIDefaultInstallAllUsers **95**
MSIFilename **95**
MSIInstallDirectory **96**
MSIManufacturer **96**
MSI-Parameter **31**
MSI-Parameter ändern **32**
MSIProductCode **96**
MSIProductVersion **96**
MSIRequireElevatedPrivileges **97**
MSIUpgradeCode **97**
MSIUseCabs **97**
NetRelaunch **66**
ObjectTypes **75**
OptionalAppLinks **91**
OutDir **70**
Parameter, die für die ##Attributes.ini-Datei gelten **66**
Parameterplatzierung **66**
PermittedGroups **72**
Protocols **70**
ReadOnlyData **87**
RegistryIsolationMode **69**
RemoveSandboxOnExit **99**
RequiredAppLinks **90**
ReserveExtraAddressSpace **87**
RetainAllIcons **83**
RuntimeEULA **67**
SandboxCOMObjects **76**
SandboxName **99**
SandboxNetworkDrives **99**
SandboxPath **100**
SandboxRemovableDisk **100**
Shortcut **88**
Shortcuts **88**
Source **88**
StripVersionInfo **84**
Struktur **66**
UACRequestedPrivilegesLevel **73**
UACRequestedPrivilegesUiAccess **73**
UpgradePath **77**
Version.XXXX **85**
VirtualComputerName **67**
VirtualDrives **78**
VirtualizeExternalOutOfProcessCOM **76**
WorkingDirectory **89**
Wow64 **68**
Package.ini-
Parameter **65**–**100**
Parameter
Einstellungen auf Ordnerebene statt auf Paketebene anwenden **24**
für MSI-Dateien **31**
für Package.ini **65**
für sbmerge.exe **49**
für thinreg.exe **29**
PermittedGroups, Auswirkung auf Application Link **46**
primärer Datencontainer
Beibehalten des Namens mit Application Sync **43**
definieren **18**
Größenauswirkungen **18**
Problembehandlung
Bereitstellung der benötigten Informationen für den Support **53**
Explorer.exe **62**
Java Runtime Environment **63**
Microsoft Outlook **61**
mit Protokoll-Monitor **54**
Projektdateien **22**
Protokollformat **56**
Protokoll-Monitor
Anhalten und Fortsetzen der Protokollierung **55**
Problembehandlungsverfahren **54**
verwenden **54**
zusätzliche Optionen **55**

R

regedit.exe, definieren **17**

S

Sandbox
definieren **101**
Parameter **98**
Speicherort **19**, **103**
Struktur **104**
Suchreihenfolge **101**

Überlegungen für aktualisierte Anwendungen **52**
sbmerge.exe
Befehle **49**
definieren **48**
Zusammenführen der Laufzeitänderungen **48**
Setup Capture-Assistent, verwenden **16**–**23**
Shell-Integration, beschränkte Funktionen mit ThinApp **12**
Skripts
Beispiel: .bat **116**
Beispiel: .reg **117**
Beispiel: Dateikopie **117**
Beispiel: Dienst **117**
Beispiel: Systemregistrierung **118**
Beispiel: Timeout **117**
Beispiel: Virtuelle Registrierung **117**
Gründe für **116**
Rückruffunktionen **115**
snapshot.exe
Beispielbefehle **107**
Beispielvorgang **107**
Erstellen der Snapshots von der Befehlszeile **105**
snapshot.ini, definieren **105**, **108**

T

Technischer Support
erforderliche Angaben für die Problembehandlung **53**
ThinApp
Aktualisieren von Anwendungen **41**
Anforderungen zur Installation und Kapselung von Anwendungen **11**
Bereitstellungsoptionen **27**
Empfehlungen für sauberen Computer **12**
in einer VMware View-Umgebung **27**
nicht unterstützte Anwendungen **12**
Ordnermakros **112**
Pakete per Stream vom Netzwerk übertragen **34**
Projektdateien durchsuchen **22**
thinreg.exe verwenden **28**
unterstützte Betriebssysteme und Anwendungen **11**
Verzeichnisdateien **13**
wird installiert **13**
thinreg.exe
Aktualisieren der Registrierungen mit Application Sync **43**
ausführen **29**
definieren **28**
mit Application Sync **28**
Parameter **29**
Starten mit MSI-Dateien **22**
Treiber, Unterstützung für **37**

U

Unterstützung
für Anwendungen **11**
für Betriebssysteme **11**

V

verschachtelte Links mit Application Link **45**
virtuelles Dateisystem
Formatierungsstufen **111**
Pfade mit Makros darstellen **112**
verwenden **111**
Verwendung von Isolationsmodi **111**
VMware View, unter Verwendung von gekapselten Anwendungen **27**
vregtool, Auflisten virtueller Registrierungsinhalte **104**